VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. Štěpán Grabovský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

EMULÁTOR PŘENOSOVÝCH PARAMETRŮ DATOVÝCH SÍTÍ
EMULATOR TRANSMISSION PARAMETERS OF DATA NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Štěpán Grabovský**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Václav Zeman, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Štěpán Grabovský ***ID:*** 145998
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Emulátor přenosových parametrů datových sítí

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a realizace síťového emulátoru, který umožní ovlivňovat přenosové parametry paketových sítí pracující s protokolovou sadou TCP/IP. V rámci práce specifikujte jednotlivé přenosové parametry vhodné pro emulaci, proveďte rozbor možných řešení a navrhněte koncepci zařízení, které by splňovalo očekávané parametry. Navrženou koncepci emulátoru realizujte do podoby funkčního vzorku.

DOPORUČENÁ LITERATURA:

[1] PELKA, T., POLÍVKA, M. Simulátor rozsáhlých sítí – SimPP. Elektrorevue - Internetový časopis (http://www.elektrorevue.cz), 2010 sv. 12, č. 6, s. 117-122. ISSN: 1213- 1539.

[2] HEMMINGER, Stephan. Network Emulation with NetEm. Canberra, Australia: Open Source Development Labs, 2005, . Proceedings of the 6th Australia's National Linux Conference (LCA2005).

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Diplomová práce se zabývá vývojem emulátoru přenosových parametrů datových sítí pracujících s protokolovou sadou TCP/IP. Práce obsahuje specifikaci přenosových parametrů a rozbor současných proprietárních i otevřených systémů. Dále je popsána realizace emulátoru, který jako své jádro využívá nástrojů tc, netem a iptables. Obslužná část je realizována webovou službou a pro interakci s obsluhou emulátoru bylo vyvinuto webové uživatelské rozhraní. V závěru práce jsou vypsány a následně diskutovány naměřené výsledky schopností emulátoru.

KLÍČOVÁ SLOVA

Emulátor síťových parametrů, netem, propustnost, webová služba, zpoždění.

ABSTRACT

This diploma thesis deals with a development of a network emulator operating in TCP/IP computers networks. This work contains a specification of transmission parameters and an analysis of concurrent systems both proprieties and open-sources. Afterward describes a relalization of the emulator itself. The core of the emulator uses well-known Linux tools tc, netem and iptables. An API of the program is realized by web service and for user interaction has been developed a web user interface. At the end of this work are listed and then discussed measured results of the emulator capabilities.

KEYWORDS

Netem, network emulation, web service, throughput, delay.

GRABOVSKÝ, Štěpán *Emulátor přenosových parametrů datových sítí*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 99 s. Vedoucí práce byl doc. Ing. Václav Zeman, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že Prohlašuji, že svou diplomovou práci na téma „Emulátor přenosových parametrů datových sítí“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 25. 5. 2016 .....

Štěpán Grabovský

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Václavu Zemanovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno 25. 5. 2016 .....

Štěpán Grabovský

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno 25. 5. 2016 .....

Štěpán Grabovský

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

GiTy, a.s.
Mariánské náměstí 1
617 00 Brno – Komárov
Czech Republic
www.gity.eu

PODĚKOVÁNÍ

Část této diplomové práce byla realizován ve spolupráci s firmou GiTy a.s., za což této firmě a jejímu zaměstnanci panu Ing. Martinu Sýkorovi, patří poděkování.

Brno 25. 5. 2016 .....

Štěpán Grabovský

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Při vývoji aplikací komunikujících přes počítačovou síť, správě počítačových sítí nebo při výuce vzniká potřeba sledovat chování síťových systémů v podmínkách jak ideálních, tak ztížených či rovnou kritických.

Testování počítačových aplikací a protokolů ve ztížených či proměnných přenosových podmínkách je nutné jak z pohledu uživatelského komfortu, tak z pohledu bezpečnosti. Správci podnikových sítí potřebují testovat funkčnost a chování síťových prvků, nastavení sítě, simulování nouzových stavů v síti, dostupnost kritických uzlů a dále. Změnou přenosových parametrů lze také určit, jaké minimální vlastnosti musí splňovat připojení od poskytovatele připojení do sítě Internet a šetřit tak finance. Možnost ovlivňovat přenosové parametry najde své uplatnění také ve výuce a výzkumu.

Ve výše popsaných prostředích (IT společnosti, vysoké školy, atp.) jsou ve většině případů datové sítě s nadprůměrně dobrými přenosovými vlastnostmi, které je pro testování či zkoumání nutné uměle měnit – zhoršit. Zmíněnou změnu přenosových vlastností dokáží zajistit speciální síťové prvky, tzv. síťové emulátory.

Emulátor přenosových parametrů je počítač připojený do datové sítě, jenž různým zacházením s datovým tokem, který přes něj prochází, dokáže měnit přenosové vlastnosti spoje. Je schopen snižovat počet odbavených datových jednotek za časový úsek a tím snižovat propustnost sítě, pozdržovat datové jednotky, čímž simuluje zpoždění, měnit jejich pořadí, filtrovat je dle zadaných parametrů a mnoho dalšího. Záleží jen na požadavcích uživatele.

Emulátor vyvíjený v této práci je součástí projektu č. VI20152018002 – Zátěžový tester ICT, který je řešen v rámci bezpečnostního výzkumu MVČR firmou GiTy, a.s. a Ústavem telekomunikací FEKT VUT v Brně. Kromě zmíněného emulátoru by měl výsledný produkt také testovat chování lokálních sítí a systémů při vysokém zatížení, testovat odolnost systémů proti útokům na síťovou infrastrukturu či sloužit jako síťová sonda.

Cílem této práce je návrh a realizace síťového emulátoru, který umožní ovlivňovat přenosové parametry paketových sítí pracující s protokolovou sadou TCP/IP. V následujícím textu je nejdříve věnována pozornost popisu přenosových parametrů a upřesnění dalších podstatných pojmů. Dále je představen rozbor možných řešení a na základě této analýzy popsán zvolený koncept emulátoru, za nímž následuje popis samotné realizace zařízení. V závěru práce jsou poté shrnuty výsledky měření vyvinutého emulátoru.

1 EMULACE A PŘENOSOVÉ PARAMETRY DATOVÝCH SÍTÍ

V první části kapitoly je vysvětlen pojem emulace a rozdíl mezi emulací a dalšími metodami využívanými při testování. Dále je věnována pozornost popisu a definici jednotlivých přenosových parametrů a jejich význam při emulaci.

1.1 Emulace a ostatní experimentální metody

Jak bylo naznačeno v úvodu práce, síťový emulátor je nástroj, který modeluje podmínky reálných sítí v laboratorním prostředí. Dá se také říci, že přenáší globální sítě WAN do lokálních sítí LAN.

Pro popis reálného síťového prostředí v prostředí experimentálním lze využít tří základních metod a to simulace, testovaní v reálném prostředí a kompromisní metody – emulace. Poznatky uvedené v této sekci jsou čerpány z [1], [2] a [3].

1.1.1 Simulace

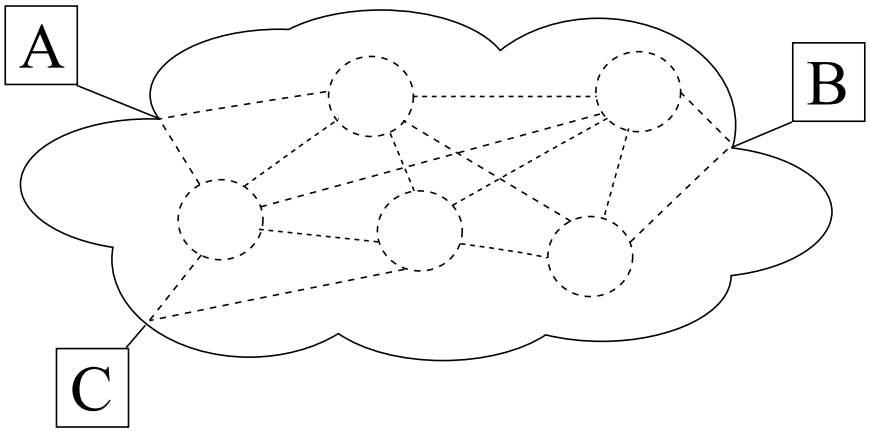
Simulace, někdy také nazývána jako počítačový model, je experimentální technika, ve které jsou fyzické komponenty systému reprezentovány pouze virtuálními komponentami v počítačové reprezentaci systému. Pro virtualizaci systémů může simulace využívat analytické modely, což jsou popisy reálných systémů pomocí matematických nástrojů. Jinými slovy, simulace modeluje vnitřní uspořádání cílového reálného systému za účelem poskytnout co nejvěrnější obraz reality, k čemuž využívá pouze softwarové prvky.

Simulace zpravidla nevyžaduje speciální hardwarové vybavení, což může výrazně snížit finanční náklady této experimentální techniky. Nevýhodou tohoto přístupu je skutečnost, že pro popis reálného systému je nutné reálné prostředí zobecnit. Jako příklad síťového simulátoru lze uvést systém ns-3.

Schématická ukázka simulace je na obrázku 1.1, všechny vyobrazené komponenty jsou virtuální.

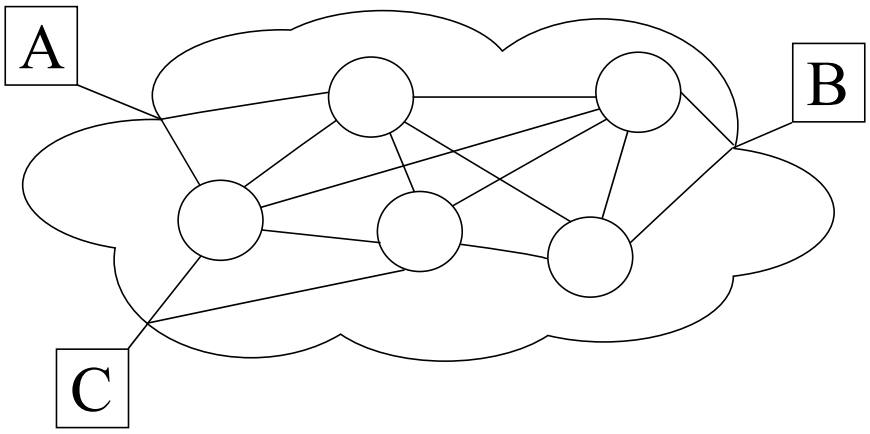
1.1.2 Testovaní v reálném prostředí

Testování v reálném prostředí je experimentální technika, která využívá pouze reálné systémy. Tyto systémy však mohou být odděleny od již pracujících reálných systémů. V takovémto případě se pro ně používá označení testovací platforma (anglicky testbed). Mezi nejznámější síťové testovací platformy patří PlanetLab, Emulab nebo OneLab.



Obr. 1.1: Simulace, všechny komponenty jsou virtuální [1].

Schématická ukázka reálného prostředí je na obrázku 1.2, všechny komponenty jsou reálné.



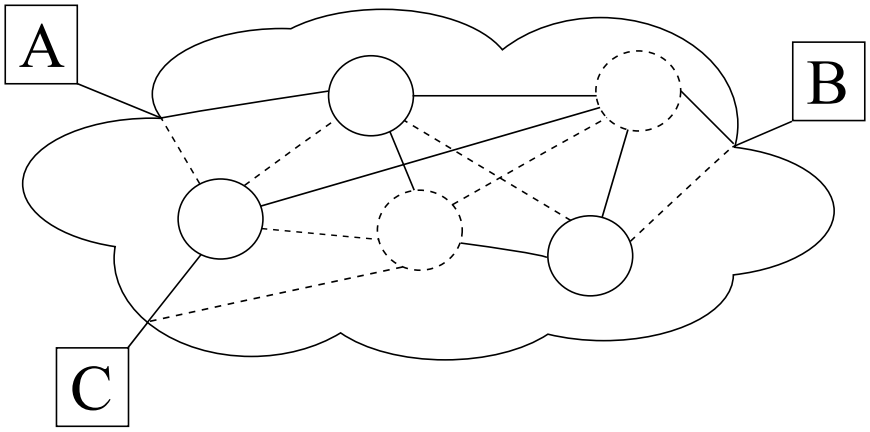
Obr. 1.2: Reálné prostředí, všechny komponenty jsou reálné [1].

1.1.3 Emulace

Emulace je experimentální technika, která kombinuje prvky simulace a reálného prostředí (viz 1.1.1 a 1.1.2). Emulace je tak kompromisem mezi testováním v reálném prostředí a simulací, kde reálné prvky systému snižují míru zobecnění a virtuální části přispívají ke snížení ceny této techniky. Dá se také říci, že emulace je technika, která na vyhrazeném reálném hardwaru provozuje software (ať již reálný nebo vyjádřený analytickými modely) zajišťující napodobení skutečných vlastností cílového prostředí.

Schématická ukázka emulace je na obrázku 1.3, část komponent je reálná a část virtuální.

Emulátor je tedy zařízení, které poskytuje emulaci, tj. kompromis mezi simulací a reálným prostředím, kdy kombinuje reálné a virtuální prvky tak, aby bylo dosaženo co nejvěrnějšího napodobení skutečné předlohy s co nejnižší cenou.



Obr. 1.3: Emulace, část komponent je reálná a část virtuální.

1.2 Přenosové parametry datových sítí

Přenosové parametry jsou veličiny popisující přenosové vlastnosti datového spoje. Hlavními přenosovými parametry jsou: propustnost, zpoždění, záměna pořadí, duplikace, ztrátovost a chybovost.

1.2.1 Propustnost

Propustnost C (b/s), anglicky throughput, je veličina, která udává přenosovou kapacitu nebo také přenosový výkon spoje. Citovaná literatura ([4], [5], [6], [7], [8], [9], [11], [12], [13], [14]) definuje propustnost jako objem přenesených dat d (bitů či bloků bitů – např. paketů) za daný čas t, viz vztah 1.1.

$$C = \frac{d}{t} \qquad [\mathrm{b/s}] \tag{1.1}$$

Jednotlivé zdroje se však rozchází v pojetí přenesených dat d. Většina literatury uvažuje data d jako všechna přenesená data, tj. užitná i řídicí. Výjimkou je zdroj [14] definující propustnost C_g jako poměr pouze užitných dat g, nikoli režie, ku času t. Viz vztah 1.2.

$$C_{\mathrm{g}} = \frac{g}{t} \qquad [\mathrm{b/s}] \tag{1.2}$$

Parametr, který je definován jako poměr všech přenesených dat d za daný čas t, je autorem zdroje [14] označován jako přenosová rychlost v, anglicky bit rate.

Přenosová rychlost

Napříč všemi použitými prameny je přenosová rychlost $v\,(\mathrm{b/s})$ definována shodně jako objem přenesených dat d za daný čas t. Na rozdíl od autora pramene [14] zmíněného výše, však přenosovou rychlost pojímají buď jako synonymum k propustnosti, vztah 1.1 (prameny [9], [10] a [6]), anebo přesně opačně, jako poměr užitných dat g ku času t, viz vztah 1.2 (prameny [5] a [13]).

Přenosová rychlost definovaná jako poměr užitných dat g ku času t je někdy označovaná jako goodput.

Šířka pásma

Dalším pojmem používaným pro označení propustnosti je šířka pásma $B\,(\mathrm{b/s})$, anglicky bandwidth. Tato veličina se primárně používá při analogovém přenosu dat pro označení kmitočtového rozmezí a její jednotkou je Hz. Odtud začala být přeneseně používána i při digitálním přenosu dat ovšem s jednotkami b/s (stejně jako propustnost 1.1).

Ve většině zkoumaných pramenů je šířka pásma brána jako synonymum k propustnosti nebo jako všeobecně dané dogma (např. [9], [10] nebo [4]). Rozdíl mezi propustností a šířkou pásma je exaktně definován pouze ve zdroji [8] a to tak, že šířka pásma je vnímána jako teoretická hodnota daná charakteristikami přenosových spojů a uzlů a je tedy vyšší než propustnost.

Za zmínku stojí také tvrzení uvedené v [5], kde i přes název digitální šířka pásma je tato veličina pojímána pouze ze signálového úhlu pohledu. Šířka pásma při přenosu digitálních dat má podle zmíněného zdroje nekonečnou velikost a je udávaná v Herzích, nikoliv b/s.[1] Pojem šířka pásma s jednotkou data za čas tak tento zdroj neuznává a nahrazuje jej plně pojmem propustnost.

Souhrn

Z výše popsaného rozboru je patrné, že výběr vhodného výrazu pro určení kapacity není jednoznačný, tím spíše, že zmíněný rozbor čerpá pouze z odborné literatury. V odborných článcích, technických zprávách nebo katalogových listech emulátorů je

[1] Digitální (obdélníkový) signál má ve frekvenční oblasti nekonečně mnoho vyšších harmonických vln, tj. nekonečně mnoho frekvencí.

používání výše zmíněných pojmů ještě více zavádějící. Obecně bývá autory nejčastěji volen výraz šířka pásma, méně častěji pak přenosová rychlost a propustnost.

V této diplomové práci bylo čeleno dilematu, zdali používat termín, který není termínem zcela vhodným, ovšem je v prostředí emulace hluboce etablován, anebo zda používat názvosloví, které je formálně správné, ovšem v reálných případech používané jen zřídka. Vzhledem k charakteru práce a také na základě výše zmíněných důvodů byl nakonec vybrán pojem propustnost. Dalšími důvody byly skutečnosti, že výraz šířka pásma má autor textu spojen spíše s rozdílem kmitočtů u rádiového přenosu a i z podstaty emulace tento pojem příliš nereflektuje základní principy emulátoru. Emulace propustnosti, jak je detailněji popsáno dále 2.1, je realizována řízením fronty, která na základě přidělování žetonů *propouští* datové jednotky ven do síťového rozhraní emulátoru.

1.2.2 Zpoždění

Zpoždění, anglicky delay nebo latency, je veličina, která vyjadřuje časový rozdíl mezi vysláním přenášené informace (bit, bajt, buňka, rámec, paket, ...) zdrojem a jejím přijetím v přijímači [15]. Zpoždění je závislé na délce trasy, velikosti přenášené zprávy, propustnosti, provozu a výkonu jednotlivých uzlů v systému. Zpoždění při přenosu mezi stanicemi v jedné lokalitě se pohybuje okolo 1 ms, zatímco přenos mezi kontinenty pak může trvat až 100 ms. Zpoždění lze dělit do tří základních kategorií dle příčiny nebo místa vzniku, místa měření a změny v čase.

Dle příčiny nebo místa vzniku

V této kategorii lze dále definovat čtyři typy zpoždění.

- Zpoždění zpracování (anglicky Processing Delay) – zahrnuje čas potřebný ke zpracování datové jednotky, kontrole její správnosti, rozhodnutí o dalším postupu atp.
- Zpoždění front (anglicky Queuing Delay) – označuje čas, který datové jednotky stráví v systému front jednotlivých uzlů sítě.
- Přenosové zpoždění (anglicky Transmission Delay) – popisuje čas, který trvá uzlu sítě poslat (vložit) datovou jednotku na síťové médium.
- Zpoždění signálu (anglicky Propagation Delay) – je čas potřebný pro šíření signálu v médiu od zdroje k cíli.

Celkové zpoždění spoje d_{spoje} je pak dáno součtem všech výše popsaných zpoždění:

$$d_{\text{spoje}} = d_{\text{zpracovani}} + d_{\text{fronty}} + d_{\text{prenosu}} + d_{\text{signalu}} \qquad [\text{s}]. \tag{1.3}$$

Z pohledu emulace nemá toto rozdělení příliš velký význam a bere se obecně jako čas potřebný pro překonání vzdálenosti z výchozího do cílového bodu měření.

Detailní rozbor lze nalézt například zde [11]. Zmíněný výchozí a cílový bod může být rozdílné anebo i stejné místo v topologii sítě, jak je popsáno níže.

Dle místa měření

Zde existují dva typy zpoždění a to zpoždění „od konce ke konci“ neboli jednosměrné zpoždění (anglicky end-to-end) a obousměrné zpoždění (Round-Trip-Time – (RTT), které vyjadřuje souhrnné zpoždění trasy tam a zpět [11][16].

Jednosměrné zpoždění $d_{\text{jednosmer}}$ je rovno součtu zpoždění všech spojů d_{spoje} (viz 1.3) na trase od zdroje k cíli, tak jak je znázorněno ve vztahu 1.4, kde N udává počet spojů na trase.

$$d_{\text{jednosmer}} = N * d_{\text{spoje}} \qquad [\text{s}] \tag{1.4}$$

Obousměrné zpoždění d_{obousmer} je pak součet jednosměrného zpoždění od zdroje k cíli d_{tam} a jednosměrného zpoždění od cíle nazpět ke zdroji d_{zpet}. Zpoždění d_{tam} a d_{zpet} mohou být různé, přičemž ve většině případů tomu tak je.

Při emulaci zpoždění se emuluje jednosměrné zpoždění, poněvadž emulátor je aplikován vždy na daný síťový tok, tj. na dané síťové rozhraní a na daný směr. Pro nastavení obousměrného zpoždění je nutné přidat další emulační podmínku pro opačný směr anebo zapnout speciální mód zrcadlení.

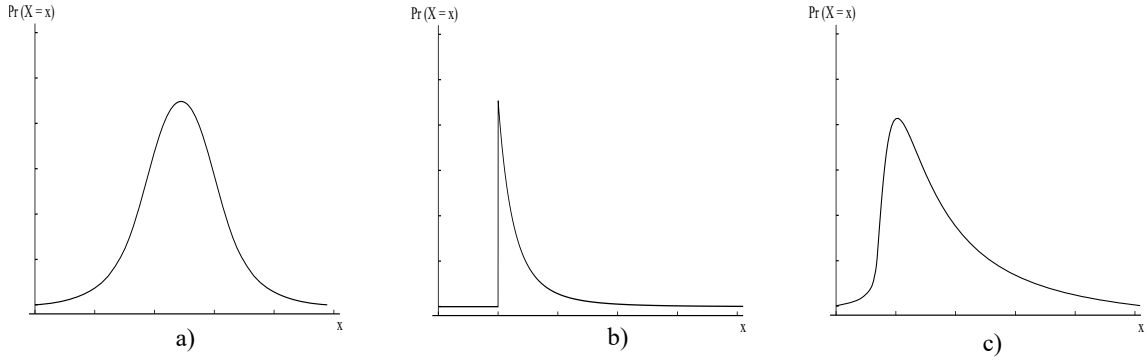
Dle změny v čase

Proměnný typ zpoždění se označuje jako chvění či kolísání zpoždění, ovšem i v českém jazyce je běžně používán anglický termín Jitter [15]. Jitter $J\,(\text{s})$ je změna velikosti zpoždění v čase, tj. odchylka aktuálního zpoždění d_{o} od střední anebo konstantní hodnoty zpoždění d:

$$J = |d - d_{\text{o}}| \qquad [\text{s}]. \tag{1.5}$$

Jak bylo naznačeno výše, jitter je způsoben faktem, že každý fragment jedné zprávy může cestovat od vysílače k cíli odlišnou cestou, která může vést přes jiné médium, jiné mezilehlé uzly, může být jinak řazena do front atp. Cílový přijímač proto nepřijímá jednotlivé části zprávy se stejným časovým odstupem. Na tento jev jsou citlivé aplikace přenášející data v reálném čase (např. VoIP, VVoIP, IPTV či VoD). Systém, který se vyrovnává s problémem proměnného zpoždění, se nazývá buffer. Jedná se o paměťový prostor, do kterého jsou přijaté části zprávy uloženy, poskládány dle správného pořadí a teprve poté předloženy nadřazené aplikaci či uživateli. Určení velikosti vyrovnávací paměti není jednoduché a vždy se jedná o kompromis mezi časovou přesností, velikostí paměti a uživatelským komfortem. Z tohoto důvodu je jitter jeden z nejdůležitějších parametrů emulace.

Jitter nabývá hodnot náležící intervalu, který je dán maximální a minimální hodnotou zpoždění. Aktuální hodnota zpoždění může být náhodná (resp. pseudonáhodná) anebo je její hodnota dána pravděpodobnostním rozložením. Nejběžnějšími typy rozložení používaných při emulaci jsou normální (Gaussovské) obrázek 1.4 a), pareto 1.4 b) a pareto-normální 1.4 c).



Obr. 1.4: Pravděpodobnostní rozložení a) normální, b) pareto a c) pareto-normal.

1.2.3 Záměna pořadí

Záměna pořadí, anglicky re-ordering, je v sítích s komutací paketů častým jevem. Podstata tohoto jevu je založena na proměnném zpoždění, kdy jednotlivé datové jednotky dosáhly během své cesty sítí jiného zpoždění, a proto mohou na síťové rozhraní cílového uzlu dorazit v jiném pořadí, než v jakém byly vyslány. Podstata záměny pořadí na základě proměnného zpoždění je vysvětlena na následujícím příkladu.

Mějme datovou jednotku, která je vyslána v čase $t_{0_{\text{start}}}$ *a po trase dosáhne zpoždění 50 ms, s odchylkou 10 ms. Její výsledné zpoždění bude*

$$50\,ms + 10\,ms = 60\,ms,$$

a tak dorazí do cílového uzlu v čase

$$t_{0_{\text{cil}}} = t_{0_{start}} + 60\,ms.$$

Za touto datovou jednotkou byla vyslána druhá datová jednotka v čase

$$t_{1_{\text{start}}} = t_{0_{\text{start}}} + 1\,ms,$$

tj. s odstupem 1 ms. Druhá datová jednotka dosáhne zpoždění 50 ms s odchylkou 0 ms. Celkové zpoždění druhé datové jednotky je

$$50\,ms + 0\,ms = 50\,ms.$$

Tato druhá datová jednotka dorazí v čase

$$t_{1_{\text{cil}}} = t_{1_{\text{start}}} + 50\,ms = t_{0_{\text{start}}} + 51\,ms,$$

což je o 9 ms dříve než první datová jednotka a tudíž došlo k záměně pořadí.

Další příčinou záměny pořadí může být ztráta datové jednotky a její opětovné odeslání.

Záměna pořadí $R\,(\%)$ je definována jako procento datových jednotek doručených v nesprávném pořadí r z celkově přenesených datových jednotek s:

$$R = \frac{r}{s} * 100 \qquad [\%]. \tag{1.6}$$

1.2.4 Ztrátovost

Ztrátovost datové jednotky $L\,(\%)$ (anglicky Packet Loss) vyjadřuje procento ztracených (nejčastěji zahozených) datových jednotek l ze všech vyslaných datových jednotek s, viz vztah 1.7. Ztráta datové jednotky může nastat z důvodu zahlcení směrovače a nemožností směrovače obsloužit v jeden okamžik velké množství požadavků [17].

$$L = \frac{l}{s} * 100 \qquad [\%] \tag{1.7}$$

Při emulaci ztrátovosti v reálném čase ovšem není předem známé, kolik datových jednotek bude odesláno, a proto není možné určit, kolik a které datové jednotky budou emulátorem zahozeny. Princip emulace ztrátovosti je tedy poněkud odlišný. Procento ztrátovosti nastavované na emulátoru neurčuje, kolik datových jednotek bude zahozeno, ale s jakou pravděpodobností bude každá jedna datová jednotka zahozena. Tato skutečnost zvyšuje chybu emulace ztrátovosti, která se však s rostoucím počtem přenesených jednotek asymptoticky přibližuje nule.

1.2.5 Chybovost

V kontextu tohoto textu je chybovostí $p\,(\text{-})$ uvažována veličina, která popisuje kvalitu fyzického spoje, tzv. bitová chybovost (BER – Bit Error Rate) [16]. Na rozdíl od výše jmenovaných parametrů nepracuje s celými datovými jednotkami, ale s jednotlivými přenášenými bity. Bitová chybovost je vyjádřena jako poměr chybně přenesených bitů b_{ch} ku počtu všech přenášených bitů b_{c}, viz vztah 1.8.

$$p = \frac{b_{\mathrm{ch}}}{b_{\mathrm{c}}}, \tag{1.8}$$

2 ROZBOR SOUČASNÝCH EMULÁTORŮ

S rozvojem Internetu a vývojem aplikací používajících počítačovou síť, tj. pracujících tzv. online, rostla i potřeba tyto aplikace testovat. Vývoj moderních síťových emulátorů tak lze pozorovat někdy na přelomu tisíciletí [1]. V současnosti existuje velké množství emulátorů, ať už v podobě komplexního produktu, tj. hardware i software, anebo jako softwarová řešení, která lze dále dělit na systémy komerční a otevřené. Poněvadž je cílem této práce postavit systém na otevřených technologiích a také proto, že architekturu a kód otevřených programů lze snadněji zkoumat, bude největší pozornost věnována právě části věnující se systémům s volnou licencí. Pro srovnání pak následuje popis vybraných komerčních produktů. Všechny systémy pracují v prostředí protokolové sady TCP/IP.

2.1 Otevřené systémy

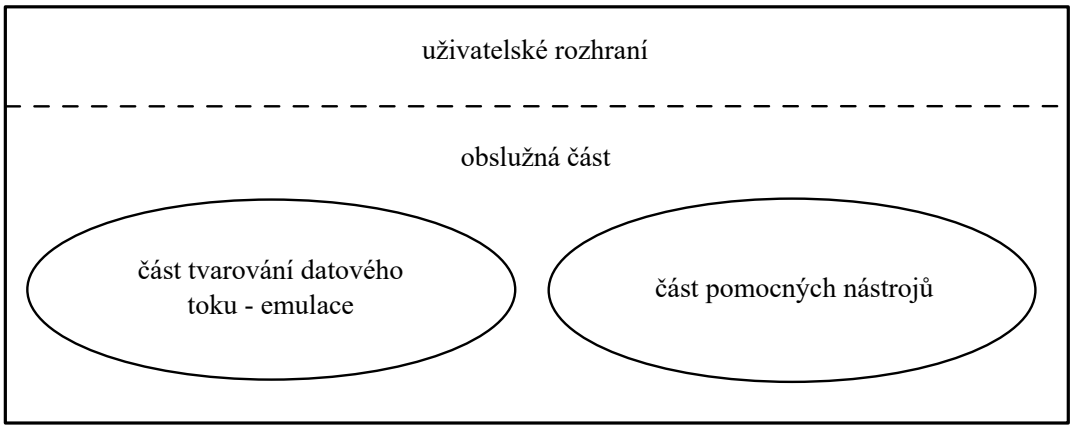
Otevřené systémy, označované také open-source, jsou řešení, jejichž autorské právo je chráněné všeobecnou veřejnou licencí GNU GPL, BSD licencí nebo podobnou volnou licencí. Tyto systémy jsou výhradně softwarové produkty, které jsou pevně implementovány do daného operačního systému či jsou samostatným programem, jejž lze do operačního systému nainstalovat.

2.1.1 Struktura open-source emulátorů

Všechny zkoumané open-source programy emulující síťové parametry jsou založeny na funkcionalitách, které jsou již součástí jádra daného operačního systému a zajišťují tak pouze jejich volání a správu. Obecný model emulátoru lze rozdělit na čtyři části a to část zajišťující tvarování datového toku, část pomocných nástrojů, část obslužnou a část uživatelského rozhraní. Situace je naznačena na obrázku 2.1.

Část tvarování datového toku

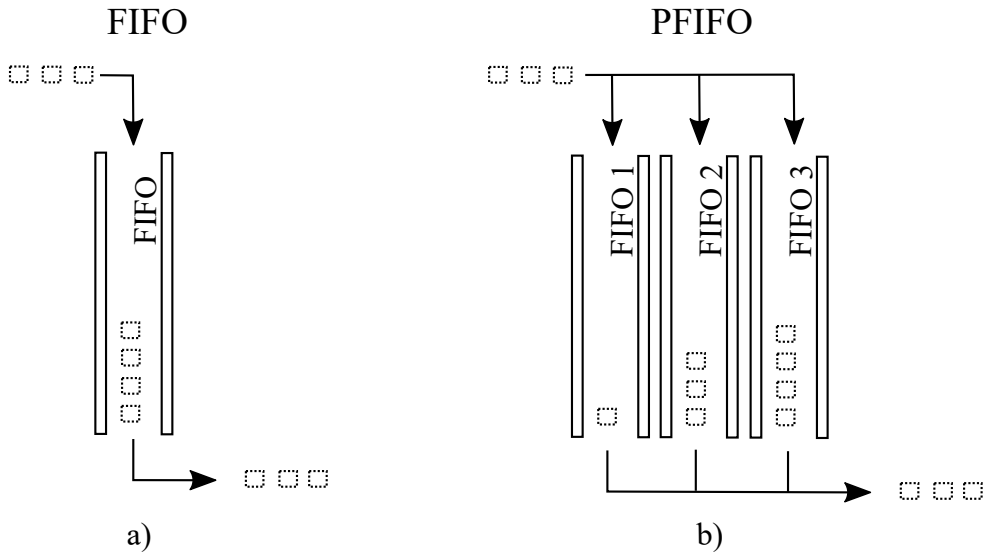
Část tvarování datového toku je jádrem emulátoru. Obsahuje systémy, které upravují datový tok a mění tím hodnoty přenosových parametrů – emulují. V současné době jsou dostupné dva programy zajišťující emulaci a to NetEm a Dummynet. Oba zmíněné nástroje jsou založeny na systému front a jsou součástí jádra svého domovského operačního systému. Tyto systémy jsou detailněji rozebrány v podsekcích 2.1.2 a 2.1.3. Třetím známým emulátorem je program NIST Net, ovšem jeho vývoj byl ukončen a většina jeho funkcionalit byla začleněna do nástroje NetEm [23].



Obr. 2.1: Obecné schéma modelu emulátoru.

Systém front nebo také řízení front (anglicky Queuing Discipline) je algoritmus, který spravuje příchozí a odchozí fronty zařízení [18]. Podle možnosti dědit vlastnosti nadřazeného řízení fronty lze rozlišovat nehierarchické (classless) a hierarchické (classful) systémy front.

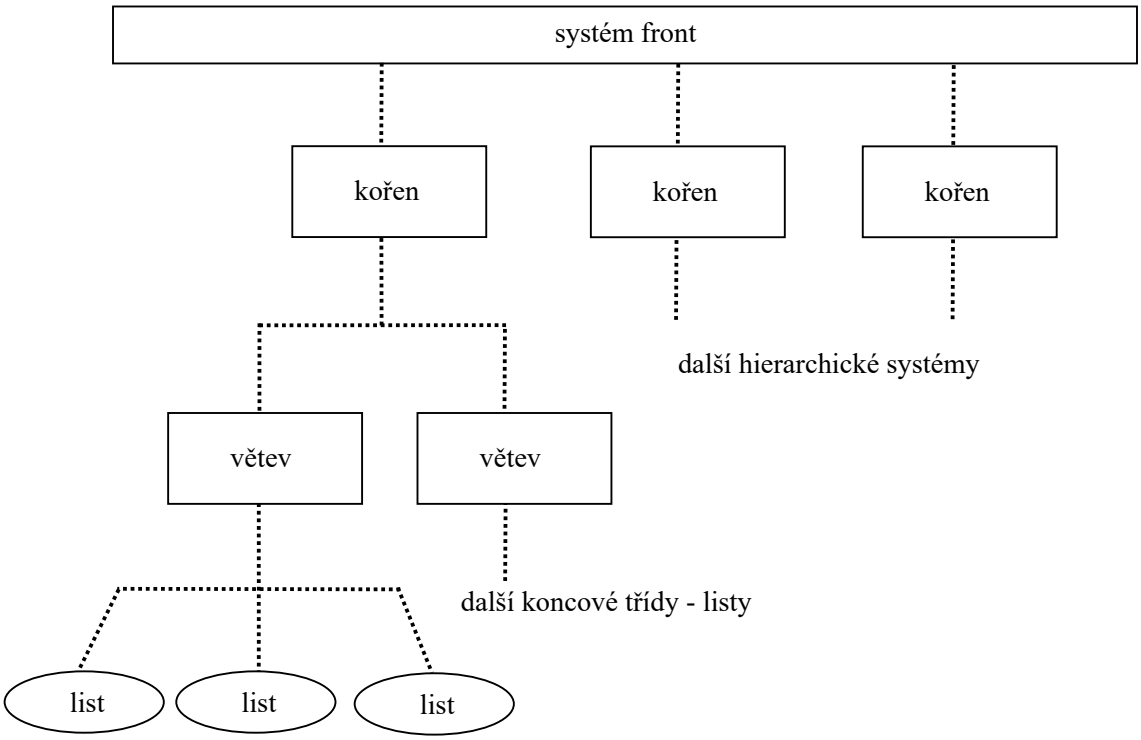
Nehierarchické systémy front jsou systémy front, které nedovolují odvozovat jednu třídu řízení front od druhé [19]. Mezi zástupce nehierarchického řízení front patří v operačním systému Linux systémy FIFO obrázek 2.2 a) a PFIFO obrázek 2.2 b).



Obr. 2.2: Systémy front FIFO a) a PFIFO b) [19].

Hierarchické systémy front jsou taková řízení front, která obsahují více tříd, jež dědí všechny parametry nadřazeného řízení a rozvíjí jej o další nastavení [18]. Pokud

třída nemá žádného potomka, je označována jako tzv. list (anglicky leaf). Hierarchické systémy vytvářejí stromovou strukturu, jak je naznačeno na obrázku 2.3. Zástupcem hierarchického řízení front v operačním systému Linux může být systém HTB (Hierarchical Token Bucket), který nahrazuje zastaralý CBQ (Class Based Queuing). Více informací o teorii a systémech front lze nalézt kromě citované literatury také v [20] [21] nebo [22].



Obr. 2.3: Hierarchické systémy front [19].

Část pomocných nástrojů

Pomocnými nástroji jsou systémy zajišťující přídavné funkce k emulaci. Mezi tyto nástroje patří především filtry, které aplikují rozdílné emulační parametry na rozdílné datové toky. U emulátorů využívajících hierarchický systém front realizují obsluhu frontové hierarchie. Dále také ovládají nastavení síťových rozhraní, upravují směrování a další pomocné aktivity. Stejně jako nástroje pro tvarování datového toku jsou již ve většině případů součástí operačního systému.

Obslužná část

Funkcí obslužné části je ovládat podřízené systémy (část tvarování datového toku a část pomocných nástrojů) a koordinovat jejich součinnost. Obslužná část dále

zprostředkovává komunikaci s uživatelským rozhraním a sbírá informace ze systémů, které zpětně uživatelskému rozhraní poskytuje. Jedná se o mezivrstvu, která nemusí být striktně ohraničena, v některých případech je přímou součástí uživatelského rozhraní. Příkladem obslužné části může být aplikační rozhraní, tzv. API, anebo kontroler z návrhového vzoru MVC.

Část uživatelského rozhraní

Část uživatelského rozhraní zajišťuje interakci s lidským uživatelem a zpětně prezentuje uživateli stav programu. Uživatelské rozhraní může být konzolové, grafické nebo webové. Častá je i jejich kombinace.

2.1.2 NetEm

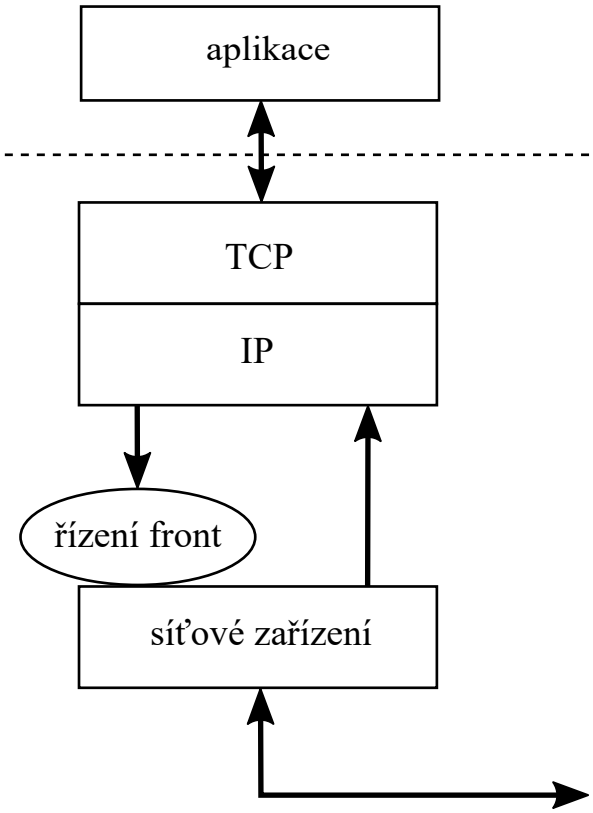
NetEm je program rozšiřující jádro operačního systému Linux. Jeho autorem je Stephan Hemminger a součástí jádra Linuxu je od verze 2.6 jako část kolekce balíčků iproute2, konkrétně nástroje Traffic Control. NetEm se skládá ze dvou částí, komunikačního modulu Netlink pro konfiguraci a modulu pro řízení front.

Modul Netlink socket interface je část jádra linuxového operačního systému, která je používána pro meziprocesovou komunikaci (IPC), jež zprostředkovává výměnu informací mezi jádrem systému a uživatelskými procesy anebo mezi uživatelskými procesy (procesy jednotlivých uživatelských aplikací) navzájem. Celá komunikace je podobná komunikaci v počítačových sítích, má své datové jednotky – sokety, které jsou označeny adresou, v tomto případě PID. Netlink je navržen tak, aby byl schopen přenášet jakákoliv data z počítačové sítě mezi jádrem a uživatelem, čehož využívají především moduly balíčku iproute2, včetně NetEm.

Řízení front je v nástroji Traffic Control umístěno na výstupu ze zásobníku síťových protokolů a na vstupu do síťového rozhraní, tj. parametry toku jsou upravovány na výstupu ze systému. Pozice řízení front je znázorněna na schématu 2.4. Řešení NetEm využívá hierarchického řízení front, které spojuje tak, aby bylo dosaženo žádaných změn u síťových parametrů. Tento soubor řízení front je pak v příkazech označován jako netem a za ním následují zadané parametry pro tvarování provozu, jak je znázorněno na ukázce pro jednoduché nastavení zpoždění.

```
1 # tc qdisc add dev eth0 root netem delay 100ms
```

V předchozích odstavcích bylo čerpáno z [24] a [25].



Obr. 2.4: Pozice řízení front v OS Linux [24].

Funkcionalita NetEm

Současná verze NetEm nabízí tyto funkcionality [25][26].

- Zpoždění – konstantní i jitter, jehož náhodnost lze ovlivnit pomocí korelace anebo pravděpodobnostními rozloženími.
- Ztrátovost – jedná se o zahazování paketů a to buďto náhodně, podle čtyřstavového Markovova modelu, nebo dle modelu Gilbert-Elliot a jeho speciálních případů Gilbertova a Bernoulliho modelu.
- ECN – je alternativou k zahazování paketů, pakety nejsou zahozeny, ale pouze označeny.
- Chybovost – nabízí emulaci chyby v zadaném procentu paketů na náhodné pozici.
- Duplikace – duplikuje pakety před tím, než jsou zařazeny do fronty.
- Záměna pořadí – zaměněné pakety lze vybírat na základě procentuálního vyjádření také společně s korelací.
- Propustnost – umožňuje měnit propustnost na základě zadaných parametrů. Kromě rychlosti lze nastavit režii linkové vrstvy, komprimaci či úplné odstranění ethernetového záhlaví nebo emulaci buněk a jejich záhlaví, například při

emulaci ATM sítě.

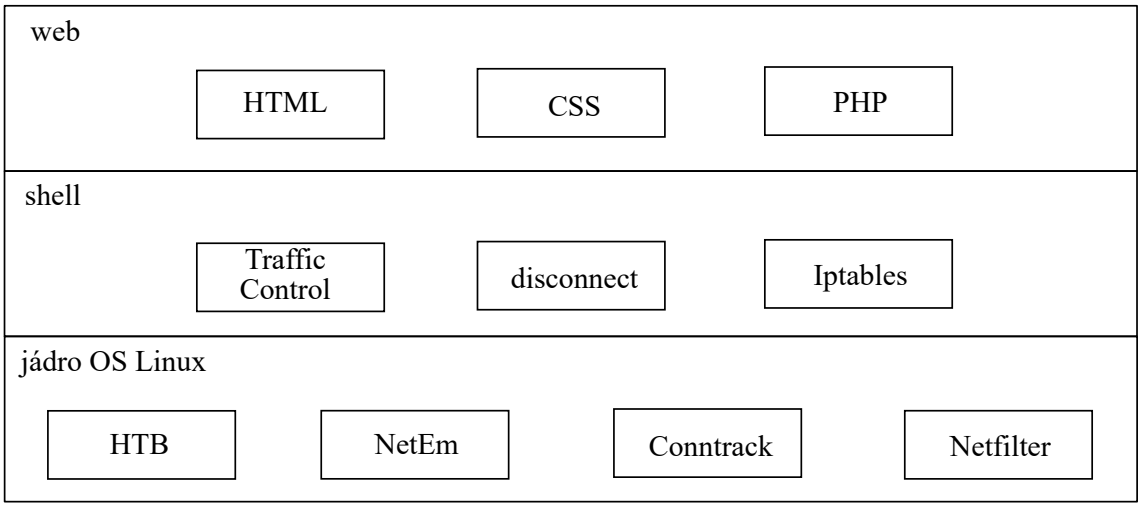
Systémy využívající NetEm

Mezi nejznámější otevřená řešení využívající NetEm patří systémy WANem, Comcast a NetPath.

WANem

WANem je emulátor globální sítě vyvíjený týmem vývojářů z Performance Engineering Research Centre společnosti TATA Consultancy Services v Indii pod licencí GNU GPL v2. WANem plně využívá funkcionalit NetEm, případně tc, tudíž dovoluje emulovat zpoždění, ztrátovost, chybovost a další výše zmíněné parametry. Navíc obsahuje možnost emuluce výpadků spojení. Program je pevně implementován do linuxové distribuce Knoppix a distribuován jako instalační obraz. Pro jeho spuštění stačí zavést upravenou distribuci Knoppix do síťového uzlu a směrovat síťový provoz přes tento uzel. Fungování programu je založeno na principu generování skriptu vytvořeného na základě uživatelem nastavených parametrů ve webovém uživatelském rozhraní. Získaný skript je poté jednoduše spuštěn.

Technologie, které byly při realizaci použity, jsou znázorněny na obrázku 2.5 a jejich popis následuje pod zmíněným obrázkem.



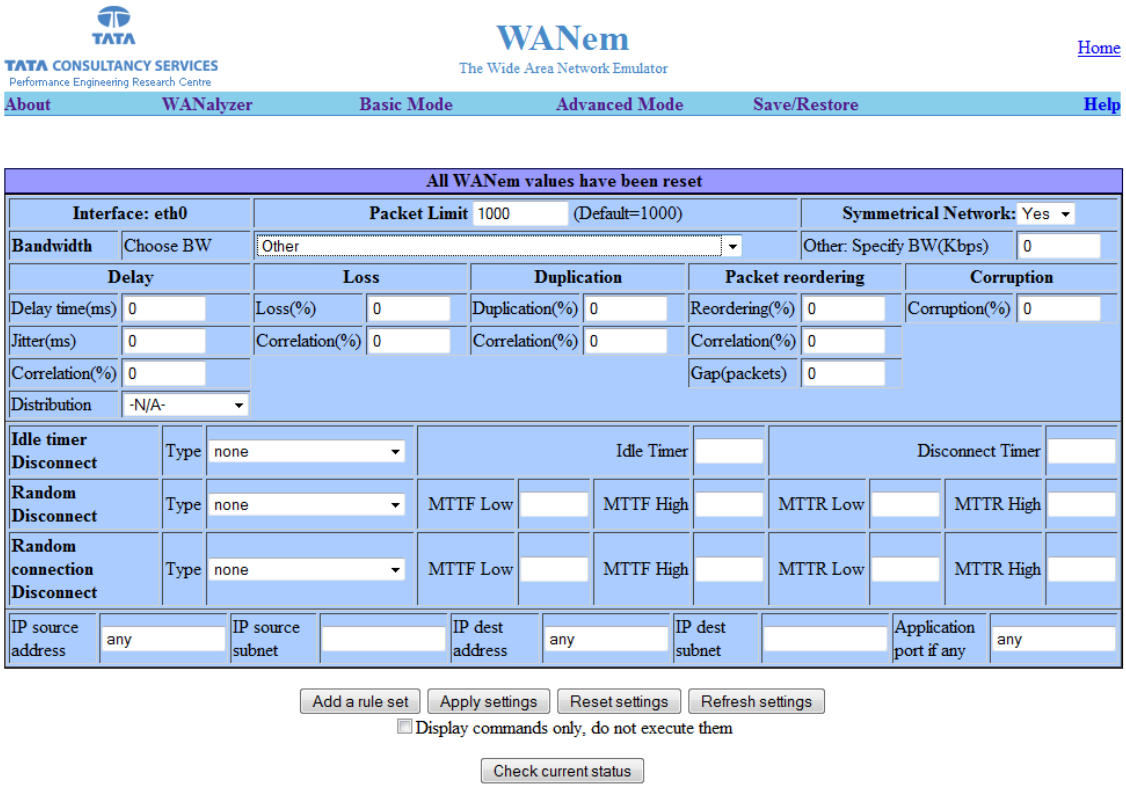
Obr. 2.5: Architektura emulátoru WANem [28].

- NetEm a Netfilter již byly popsány výše.
- HTB – Hierarchical Token Bucket je řízení fronty, které je implementováno v jádře operačního systému Linux.

- Conntrack – představuje modul v jádře operačního systému sloužící ke sledování TCP/IP spojení.
- Disconnect – je skript vyvinutý pro emulaci výpadků spojení.
- PHP skripty – vytváří webové uživatelské rozhraní a jsou založeny na otevřeném projektu PHPNetemGUI.

Není-li uvedeno jinak, pocházejí údaje z dokumentace k projektu WANem [28].

Ukázka webového uživatelského rozhraní je na obrázku 2.6.

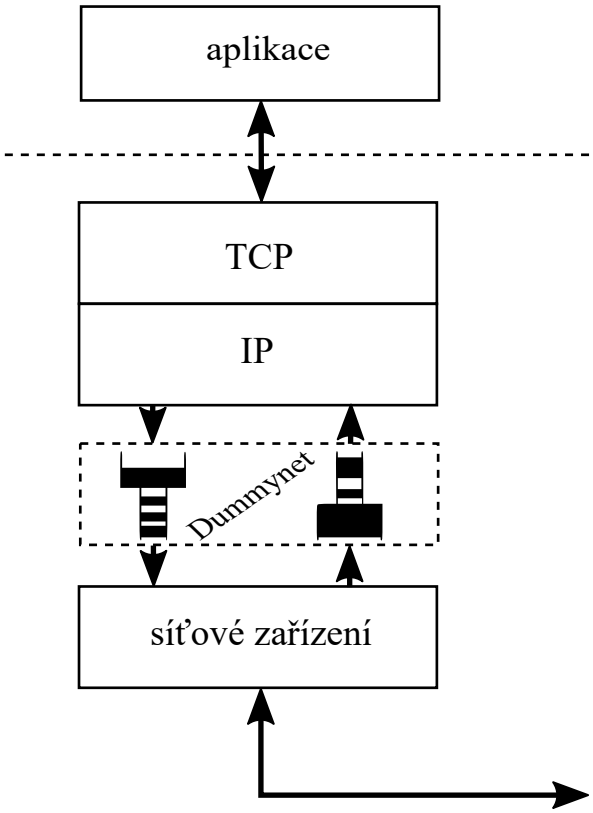


Obr. 2.6: Webové uživatelské rozhraní emulátoru WANem.

2.1.3 Dummynet

Dummynet je nástroj pro emulaci přenosových parametrů počítačových sítí, jehož vývoj začal v roce 1997 na Universita di Pisa v Itálii Luigiem Rizzotem. Nástroj byl původně napsán pro operační systémy BSD, ovšem nyní je již dostupný i pro systémy s jádrem Linux a MS Windows.

Dummynet funguje na rozhraní síťové vrstvy a vrstvy síťového rozhraní modelu TCP/IP a lze jej aplikovat jak na vstupní, tak výstupní provoz, viz obrázek 2.7.



Obr. 2.7: Schéma umístění emulátoru Dummynet.

Princip emulátoru Dummynet je založen na spojení dvou funkcionalit; front (anglicky queues) a rour (anglicky pipes) [31]. Toto spojení vychází z analogie k reálné situaci v sítích, kdy fronty zastupují směrovače a roury spoje mezi uzly. Roury tak slouží k emulaci propustnosti a přenosového zpoždění a fronty k emulaci ztrátovosti, duplikace a zpoždení front. Jinými slovy, roury mají pevně dané zpoždění a propustnost, kterou datový tok může využívat a fronty rozhodují, jakým způsobem jsou kapacity rour využívány.

Tak jako je systém NetEm (viz 2.1.2) úzce spojen s nástrojem Traffic Control, je Dummynet závislý na nástroji Ipfw. Ipfw je nástroj systémů BSD sloužící jako konzolové uživatelské rozhraní pro obsluhu firewallu, systému NAT a dalších síťových funkcionalit systému, včetně Dummynet. Dummynet využívá Ipfw pro třídění provozu, na který bude dané pravidlo aplikováno a pro konkrétní nastavení parametrů. Na ukázce aplikace Dummynet je příklad emulace sítě ADSL.

V předchozích odstavcích bylo čerpáno z [29] a [30].

```
# ipfw add pipe 3 out
# ipfw add pipe 4 in
# ipfw pipe 3 config bw 1Mbit/s queue 10 delay 100ms
# ipfw pipe 4 config bw 8Mbit/s queue 30 delay 100ms
```

Funkcionalita Dummynet

Výše zmíněné roury i fronty jsou v systému Dummynet realizovány systémem front. Z tohoto důvodu je možné rozdělit funkcionality emulátoru Dummynet na parametry konfigurovatelné u rour a parametry nastavitelné u front. Následující popis funkcionalit je zestručněný seznam z manuálu dostupného zde [31].
Parametry nastavitelné pro roury.

- Propustnost – umožňuje měnit propustnost na základě zadané rychlosti nebo na základě jiného zařízení, které nahradí systémové hodiny.
- Zpoždění – nastaví zpoždění pro danou rouru. Pro dosažení jitteru je nutné spojit více rour s jiným nastaveným zpožděním a jinou pravděpodobností použití dané roury.
- Název – označuje danou rouru.
- Váha – nastavuje váhu použité dané roury.
- Typ – umožňuje nastavení řízení front pro rozvržení datového provozu. Je možné aplikovat řízení front FIFO, WF2Q+, RR a QFQ.
- Profil – umožňuje nastavit více parametrů pro danou rouru a tak komplexně emulovat síťové spojení. Lze nastavit označení roury, propustnost, ztrátovost a zpoždění a jeho průběh.

Parametry nastavitelné pro roury i fronty.

- Prostor – udává velikost hešovací tabulky front.
- Maska – umožňuje přiřazení masky, dle které lze poté dynamicky měnit parametry toku v rouře.
- Fronta – upravuje nastavení fronty, tj. počet slotů nebo její rychlost.
- Ztrátovost – jedná se o náhodné zahazování paketů podle daného procenta ovlivněných paketů.
- Systém fronty – aplikuje řízení front RED (Random Early Detection) a GRED (Gentle Random Early Detection).

Systémy využívající Dummynet

Mezi nejznámější otevřená řešení využívající Dummynet patří systémy Comcast, ModelNet a KauNet.

Comcast

Comcast je otevřený projekt uživatele Tyler Treat na serveru GitHub. Jedná se o program napsaný v jazyce Go, který zjednodušuje volání příkazů emulátorů NetEm i Dummynet. Pracuje v příkazové řádce daného operačního systému a dokáže emulovat parametry sítí dle balíčku, který volá. Princip programu je založen na tvoření textových řetězců – příkazů a jejich následném volání.

```
1 # comcast --device=eth0 --latency=250 --target-bw=1000
```

2.1.4 Srovnání NetEm a Dummynet

Jak vyplývá z předchozího textu, pro emulaci přenosových parametrů jsou pro operační systém Linux dostupné dvě technologie a to NetEm a Dummynet.

První základní rozdíl obou technologií je v konceptu emulace. U NetEm probíhá tvarování síťového toku pomocí hierarchického spojování řízení front a je realizováno na výstupu z protokolového zásobníku. U Dummynet je využíváno rour a front k abstrakci reálné sítě, kde se každá roura konfiguruje zvlášť. Následně jsou roury spojovány dle aktuálních požadavků uživatele. Emulaci sítě lze u nástroje Dummynet provádět jak na vstupu, tak na výstupu síťového zařízení.

Dle [32] lze další rozdíl mezi systémy vidět v architektuře programů a stylu jejich vývoje. NetEm je neustále vyvíjen širokou komunitou, a proto obsahuje mnoho přídavků a funkcionalit. Úroveň dokumentace je však nižší a ne všechny přídavky jsou validovány. Vyžaduje také vyšší znalost prostředí emulátoru. Oproti NetEm je Dummynet vyvíjen úzkou skupinou vývojářů, kde každý nový přídavek je validován a dokumentován a jeho ovládání je méně náročné. Nevýhodou však je relativní uzavřenost systému.

Samotná schopnost přesné emulace daných parametrů byla ověřována v těchto publikovaných pracích: [33] z roku 2009, [32] z roku 2011 a [34] z roku 2014. Srovnání schopnosti emulace a dalších vlastností je v tabulce 2.1.

Tab. 2.1: Srovnání emulátorů NetEm a Dummynet.

	NetEm	Dummynet
Vyvíjen a podporován	ano	ano
Platforma	Linux	FreeBSD, Linux, Windows
Pozice emulace	výstup	vstup, výstup
Propustnost	ano	ano
Zpoždění	ano	ano
Jitter	ano	částečně
Ztrátovost	ano	ano
Poškození	ano	ne
Duplikace	ano	ne
Změna pořadí	ano	ano
Přerušení spojení	ne	ne
Abstrakce topologie	ne	ano

2.2 Komerční systémy

Pro určení základních i pokročilých parametrů vyvíjeného systému je nezbytný průzkum nabídek komerčního trhu. Analýza zpoplatněných řešení je z principu povahy těchto produktů obtížnější než u systémů s otevřeným kódem. V některých případech je obtížné zjistit i cenu zařízení či programu, která bývá součástí obchodní strategie a není tedy veřejně publikovatelná, nebo se odvíjí od konkrétního řešení pro daného zákazníka. Komerční produkty lze rozdělit na komplexní systémy a systémy pouze softwarové, jejichž popis je v následujících podkapitolách.

2.2.1 Komplexní systémy

Komplexní komerční systémy jsou hardwarové produkty určené speciálně pro emulaci parametrů v počítačových sítích. Jedná se o zařízení obsahující dvě a více síťových karet, jejichž pomocí je připojováno do lokální sítě. Řešení, které je cílem této práce, je ve finálním stavu také komplexním produktem, a proto je vhodné této části věnovat prostor.

Z technických důvodů nebylo možné pořízení ani zapůjčení komplexních produktů pro detailnější testování, a proto je provedený průzkum založen pouze na dokumentaci produktů získané z webových stránek a prospektů zaslaných oslovenými zástupci firem. Z nabídky dostupných hardwarových systémů byla k porov-

nání vybrána řešení firem JAR Technologies, Ltd. (UK), Apposite Technologies, Inc. (USA) a iTrinegy, Inc. (USA). Přehledné srovnání produktů je vypsáno v tabulce 2.2[1]. V tabulce je ukázáno, že současné vysoce-výkonové emulátory dokáží pracovat s propustností až do 40 Gb/s s výkonem odbavení až 36 milionů paketů za sekundu. Jsou schopny emulovat všechny základní přenosové parametry, tj. kromě omezení propustností také konstantní i proměnné zpoždění, ztrátovost, duplikaci a změnu pořadí paketů.

Zajímavou vlastností ke zkoumání je schopnost filtrování provozu. Dle technických specifikací jsou produkty firem JAR Technologies, Ltd. a Apposite Technologies, Inc., stejně jako open-source, schopny filtrovat do čtvrté vrstvy referenčního modelu ISO/OSI. Ovšem jako doplnění k tomuto filtrování nabízí systémy společnosti JAR Technologies, Ltd. speciální balíček filtrů pro audio a video typy protokolů aplikační vrstvy.

Všechny analyzované komerční produkty obsahovaly jako doplnění hardwaru bohaté uživatelské prostředí s intuitivním editorem nastavení a také zpětnou vazbu uživateli o emulaci. V některých případech s možností vykreslování grafů toku dat v reálném čase.

[1]Písmeno x v tabulce značí nemožnost získat daný údaj

Tab. 2.2: Srovnání komplexních komerčních produktů.

	JAR Techn., Ltd. 40,000/2	Apposite Tech., Inc. Netropy 40G	iTrinegy, Inc. Model 20
Zdroj	[36]	[37]	[38]
Cena (rok 2016)	x	50000 €	20000 €
Technologie			
	optika	optika	měď
Parametry			
Rozhraní	4	15	2
Propustnost	40 Gb/s	40 Gb/s	1 Gb/s
Paketů za sekundu	32 milionů	36 milionů	x
Parametry			
Směrování	ano	ano	ano
Přemostění	ano	ano	ano
Emulace propustnosti	300 b/s – 40 Gb/s	100 b/s – 40 Gb/s	1 Gb/s
Provoz na pozadí	ano	ano	ano
Změna pořadí paketů	ano	ano	ano
Duplikace	ano	ano	ano
Poškození paketů	ano	ano	ne
Ztrátovost			
Standardní	ano	ano	ano
Procentuální	ano	ne	ne
Náhodná	ano	ano	ano
Gillbert-Elliot	ano	ano	ne
Zpoždění			
Konstantní	do 30 s	do 10 s	do 10 s
Proměnné	ano	ano	ano
- schodové	ano	ne	ano
- normální	ne	ano	ne
- exponenciální	ano	ano	ne
- sinusové vlny	ano	ano	ne
Filtrování			
Linková vrstva	ano	ano	ne
Síťová vrstva	ano	ano	ne
Transportní vrstva	ano	ano	ne
Aplikační vrstva	ano	ne	ne

2.2.2 Pouze softwarové systémy

Pouze softwarové komerční produkty jsou počítačové programy emulující vlastnosti globální sítě především pro operační systémy Windows, poněvadž pro linuxuvé distribuce existují bezplatné emulátory popsané výše. Pro ukázku byly vybrány dva systémy firem ZTI Communications (Francie) a SoftPerfect Pty, Ltd. (Austrálie). Tato řešení jsou schopna emulovat všechny standardní přenosové parametry. Z pohledu této práce je zajímavé srovnání cen zmíněných produktů, které je vypsáno v tabulce 2.3. Údaje v této tabulce jsou z [39] a [40].

Tab. 2.3: Srovnání cen komerčních softwarových produktů.

(rok 2016)	SoftPerfect Connection Emulator	NetDisturb
Základní verze	1315 €	1350 €
Pokročilá verze	1769 €	1650 €

3 ŘEŠENÍ EMULÁTORU

V předchozích kapitolách byly definovány jednotlivé parametry datových sítí 1.2 a proveden rozbor současných systémů umožňujících emulaci globálních sítí 2. Na základě provedené analýzy je v této kapitole popsáno řešení systému, které je schopno definované parametry emulovat.

3.1 Rozbor zadání

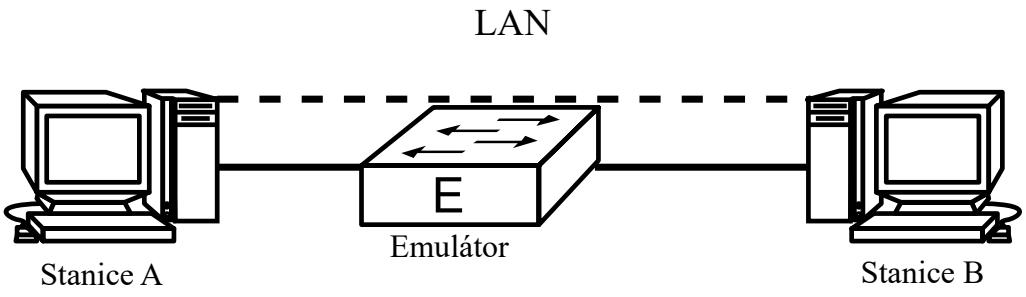
Cílem práce bylo navrhnout a poté realizovat emulátor přenosových parametrů datových sítí. Vyvíjený emulátor je součástí systému komplexního ICT testeru, který kromě emulace přenosových parametrů bude poskytovat nástroje pro zátěžové testování a síťovou sondu. Vývojovou skupinou byly stanoveny podmínky, které výsledný produkt musí splňovat anebo je na nich jinak závislý:

- celý systém ICT testeru je provozován na operačním systému Linux CentOS,
- struktura emulátoru musí být vhodně navržena, aby se v co největší míře dala použít i pro daší funkce testeru,
- ovládání emulátoru musí být dostupné přes webové uživatelské rozhraní.

V této práci ovšem nebylo řešeno technické vybavení zařízení, vývoj a testování emulátoru probíhalo na provizorním hardwaru.

3.2 Integrace emulátoru do počítačové sítě

Principiálně probíhá emulace mezi dvěma uzly sítě, kde je do komunikace mezi těmito uzly vložen mezilehlý prvek, přes který je komunikace vedena a datový tok komunikace upravován. Tak jak je naznačeno na obrázku 3.1, v němž je přerušovanou čarou znázorněn běžný tok dat a plnou čarou případ s emulací.



Obr. 3.1: Ukázka topologie emulace.

Z obrázku 3.1 vyplývá, že pro emulaci přenosových parametrů v síti je nutné zajistit, aby byl daný síťový tok veden přes síťová rozhraní emulátoru. Způsoby převedení toku dat v síti se liší dle požadavku umístění emulátoru a konkrétní topologie sítě. Problém převedení síťového toku lze dělit podle několika faktorů. Pro uživatele emulátoru je pravděpodobně nejpřívětivější celou situaci dělit na dva případy a to případ, kdy je emulace prováděna v autonomním prostředí a situaci, kdy je emulátor připojován do již existující spravované sítě. V prvním případě, případě autonomního systému, je důležité zajistit kromě samotné emulace také konfiguraci datových spojení. Naopak u připojení do již existující sítě musí emulátor co nejméně narušit již fungující nastavení.

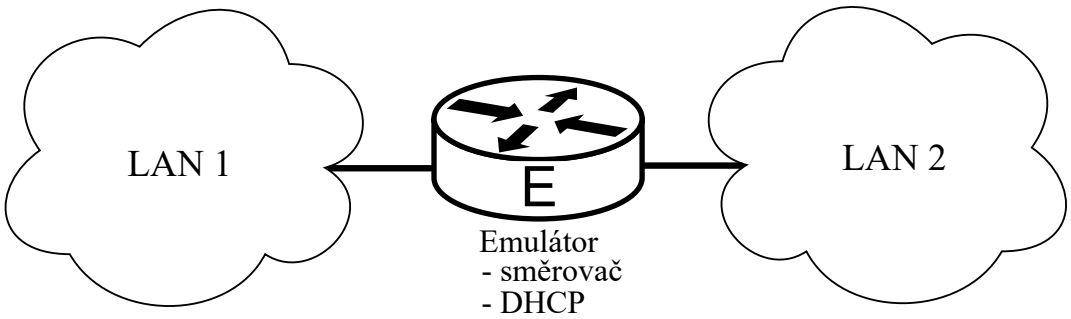
Úplný obsah konfiguračních souborů a další nastavení pro jednotlivé scénáře jsou detailně popsány v příloze A.

3.2.1 Autonomní prostředí

Jak bylo naznačeno výše, autonomním prostředím je myšlena situace, kdy je emulace prováděna v právě vytvořeném prostředí, které vzniklo za účelem dané emulace a modelování přenosových parametrů je primárním úkolem vzniklého prostředí. V autonomním prostředí zastává emulátor nejen funkci emulátoru, ale také řídícího prvku vzniklé sítě (směrovače, DHCP serveru) či výchozí brány. Tento typ zapojení sítě nachází uplatnění především v laboratorních prostředích, ať už při testování vývoje či výuce.

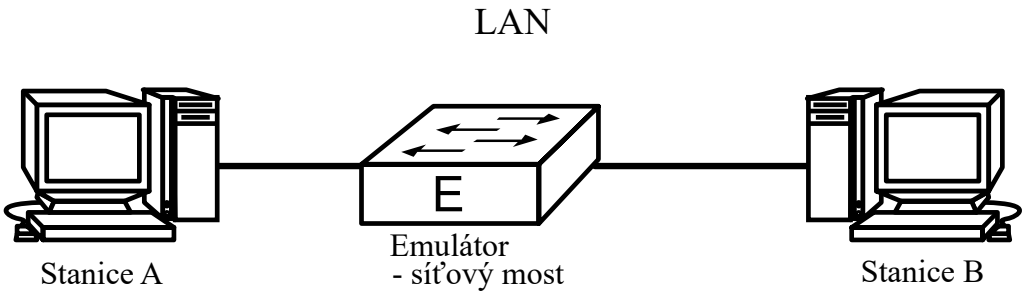
Plně autonomní prostředí

Plně autonomním prostředím se rozumí taková topologie, v níž je celá síť plně pod kontrolou emulátoru. Tento stav je naznačen na obrázku 3.2. V tomto případě je emulátor výchozí bránou pro obě sítě (LAN 1 a LAN 2) a oběma sítím také poskytuje službu DHCP, jíž jsou prvky v sítích konfigurovány.



Obr. 3.2: Plně autonomní prostředí – směrovač.

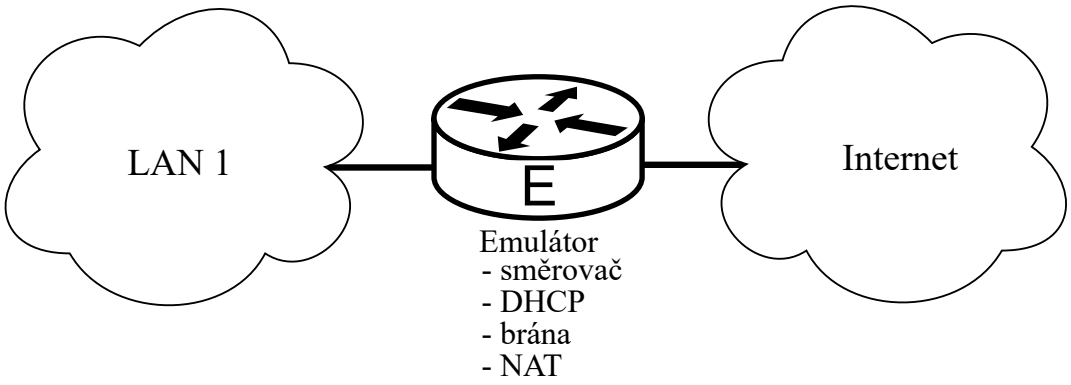
Ve výše popsaném případě se jedná o situaci, v níž emulátor funguje jako rozhraní mezi dvěma samostatnými lokálními sítěmi. Emulátor tedy pracuje na úrovni třetí – síťové – vrstvy referenčního modelu ISO/OSI. Další možností zapojení emulátoru je jako přepínač, respektive jako síťový most. V referenčním modelu ISO/OSI by se tak pohyboval na úrovni druhé – linkové – vrstvy a koncové uzly, mezi kterými by emulace probíhala, by se tak mohly nacházet ve stejné lokální síti. Výhodou tohoto zapojení sítě je skutečnost, že pro oba komunikující koncové uzly je emulátor transparentní, ovšem i přesto dokáže poskytovat službu DHCP. Vytvoření zmíněného typu zapojení je tak pro uživatele snadné a nevyžaduje žádnou úpravu nastavení koncových uzlů. Situace je znázorněna na obrázku 3.3.



Obr. 3.3: Plně autonomní prostředí – síťový most.

Autonomní prostředí s přístupem do jiné sítě

Autonomní prostředí s přístupem do jiné sítě je rozšířením plně autonomního prostředí o přístup do sítě, jejíž nastavení není pod kontrolou emulátoru, například síť poskytovatele připojení k Internetu. Emulátor tak zastupuje výchozí bránu se službou DHCP, viz obrázek 3.4. Bylo-li by k dispozici další síťové rozhraní, mohl by emulátor zároveň fungovat i jako směrovač mezi vnitřními sítěmi či jako přepínač mezi koncovými uzly, tak jak je situace známá například z domácích směrovačů.



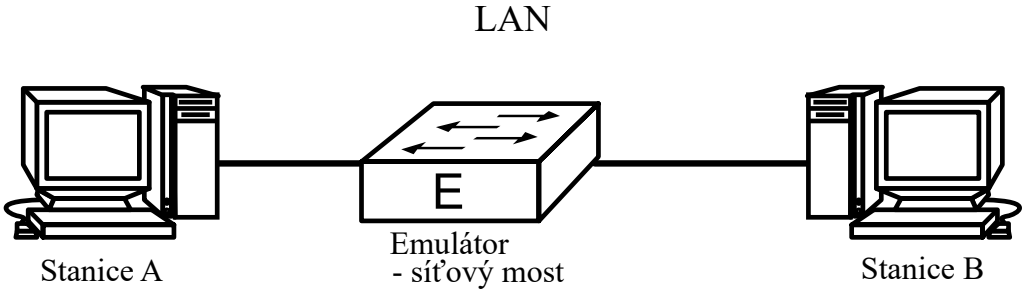
Obr. 3.4: Autonomní prostředí s přístupem do jiné sítě.

3.2.2 Připojení do existující sítě

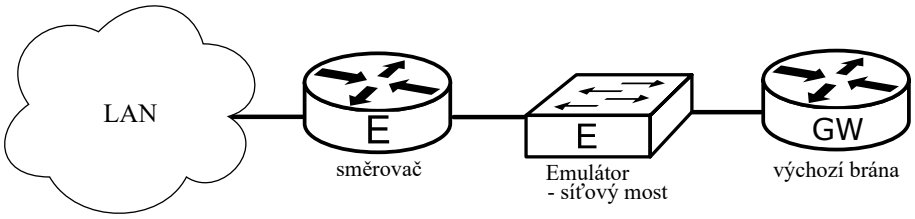
Připojení do existující sítě popisuje případy, kdy je emulátor připojován do sítě, která je již konfigurována a jejím primárním účelem není vytvoření prostředí pro emulaci. Případem takovéto situace mohou být podnikové místní sítě, které zajišťují výměnu informací jednotlivých koncových uzlů a emulace je v nich prováděna jako doplňková a dočasná činnost. Tyto sítě se zpravidla vyznačují vysokým řízením a velkým množstvím již nastavených pravidel, které komplikují zapojení prvku, jakým je emulátor. Čím složitější a více spravovaná síť je, tím větší bude muset být spolupráce se správcem sítě při integraci emulátoru.

Připojení mezi dva uzly

Připojení mezi dva uzly sítě jako transparentní síťový most je jedna z možností, jak zajistit zapojení emulátoru do již provozované místní sítě s minimální potřebou změny konfigurace uzlů v síti. Emulátor lze takto připojit mezi jakékoliv dva uzly sítě a tímto umístěním emulátoru lze určit, pro jaký rozsah sítě bude možné emulátor uplatnit. Dvě příkladné situace zapojení jsou naznačeny na obrázcích 3.5 a 3.6.



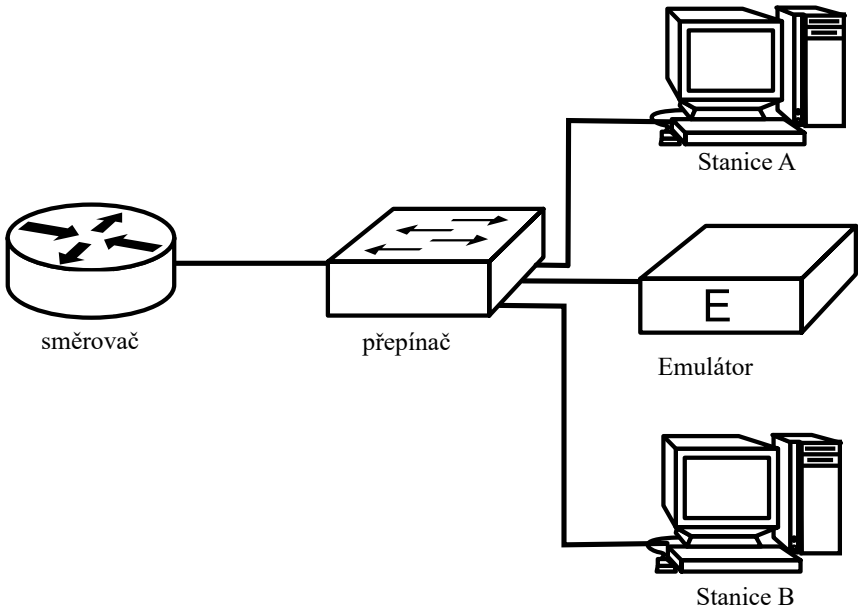
Obr. 3.5: Připojení mezi dva konfigurované uzly.



Obr. 3.6: Připojení mezi dva konfigurované uzly, směrovač a výchozí bránu.

Připojení do přepínače

Nejvíce problematické je takové připojení emulátoru do místní sítě, kdy je emulátor zapojen přímo do přepínače. Přesměrování síťového provozu tak, aby procházel emulátorem, není triviální záležitostí a je nutný zásah do přepínacích, tzv. ARP, tabulek přepínače. Situace je znázorněna na schématu na obrázku 3.7.



Obr. 3.7: Připojení emulátoru do přepínače.

Tento problém není možné řešit a v případě nutnosti použití emulátoru v takovéto situaci je vyžadován zásah administrátora dané místní sítě ke změně konfigurace přepínače. Alternativní cestou by mohlo být provedení útoku na tabulky směrovače, například tzv. ARP poisoning a přepsání konkrétních záznamů v ARP tabulce přepínače dle potřeby. Toto řešení je však na hraně morálních, ale hlavně technických možností, a to především z toho důvodu, že moderní síťová zařízení jsou již proti tomuto typu útoku chráněna.

3.3 Softwarová část emulátoru

V této sekci je popsána struktura softwaru vyvinutého emulátoru, situace struktury obecného emulátoru byla rozebrána v kapitole 2, sekci 2.1.1.

Při tvorbě jakéhokoliv softwarového produktu je vhodné využití návrhového vzoru. Návrhový vzor je předpis či šablona, která prošla určitým standardizačním procesem a která definuje strukturu programu. Návrhové vzory snižují možnost chybného rozvržení programu, umožňují jeho snadnější přenositelnost a doplnění kódu, usnadňují orientaci dalším osobám a mnoho dalších výhod.

Jelikož bylo zadáno, že emulátor bude ovládán přes webové uživatelské rozhraní, byl první volbou návrhového vzoru návrhový vzor MVC, který byl upraven tak, aby vyhovoval problému emulátoru. Obecný návrhový vzor MVC se skládá ze tří základních komponent a to datového modelu aplikace, uživatelského rozhraní a řídicí logiky [41]. Cílem návrhového vzoru MVC je co největší možná nezávislost jednotlivých částí. Jednotlivé celky tak mohou být založeny na jiné technologii a v případě potřeby snadno vyměněny. Tento návrhový vzor se v poslední době těší velké oblibě především v sektoru webových aplikací, kde v klasickém případě je část modelu zastoupena datovou databází, pohled webovou stránkou a kontrolér se stará o zprostředkování komunikace mezi těmito částmi, tj. zpracování uživatelských příkazů, získání požadovaných dat, provedení operace nad získanými daty a poskytnutí výsledků pohledové části, která je již pouze graficky prezentuje.

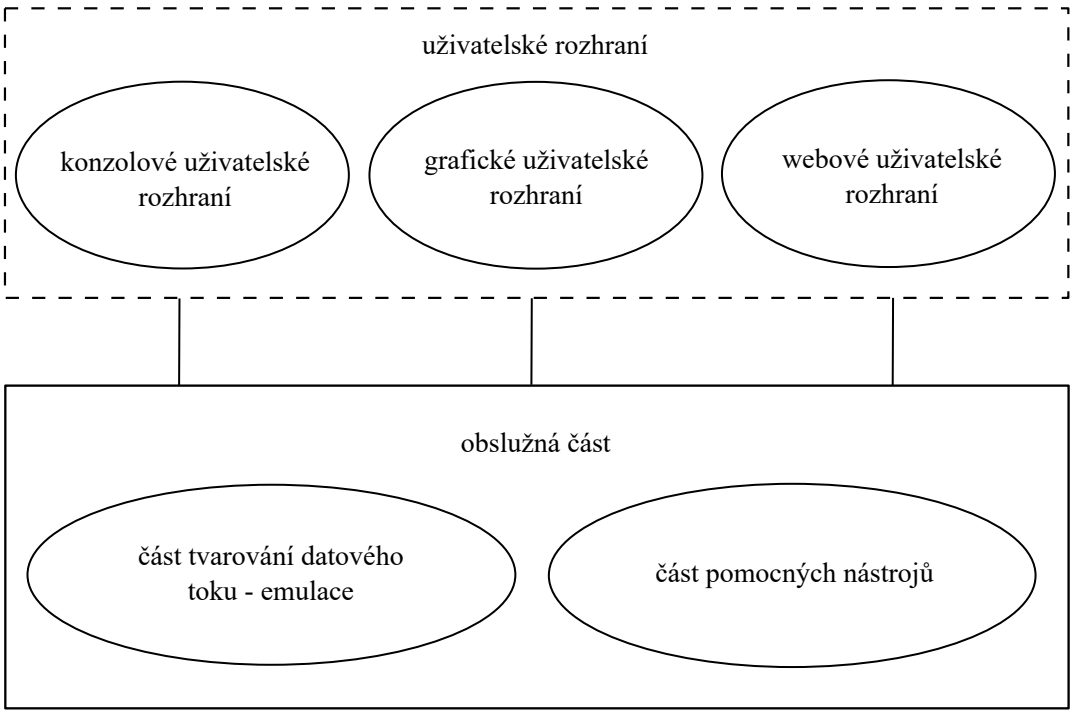
Alternativou k návrhovému vzoru MVC je struktura programu využívající tzv. aplikační rozhraní, pro něž se často používá akronym anglických slov Application Programming Interface – API. Stavba API a MVC je velice podobná a má i podobný účel, tj. oddělení jednotlivých částí programu. Speciálním případem API je tzv. Web API neboli webová služba.

Vzhledem k větší universálnosti a možnostem webového aplikačního rozhraní bylo při vývoji emulátoru upuštěno od návrhového vzoru MVC a využito struktury Web API – webové služby. Schéma stavby softwaru emulátoru je na obrázku 3.8.

V následujících podkapitolách je popsán obsah jednotlivých částí, tj. emulátorem využívaných nástrojů 3.3.1, webové služby 3.3.2 a webového uživatelské rozhraní 3.3.3.

3.3.1 Nástroj Traffic Control a Iptables

Jako část tvarování datového toku, tj. samotné emulace, viz obrázek 3.8, byl použit nástroj Traffic Control (TC) nacházející se v jádře operačního systému Linux v kolekci iproute2. Hlavním důvodem této volby jsou výsledky z testů publikovaných



Obr. 3.8: Struktura vyvíjeného emulátoru.

v [32], [33] a [34] a shrnuté v 2.1.4. Emulátor založený na Traffic Control a jeho součásti NetEm je ve srovnání s druhým dostupným emulátorem Dummynet schopný dosahovat přesnějších výsledků při emulaci šířky pásma a konstantního i proměnného zpoždění, které navíc může být charakterizováno různými křivkami pravděpodobnosti. Další výraznou výhodou NetEm je jeho snadná rozšiřitelnost anebo případná úprava kódu. Nevýhodami této volby jsou především náročnější konfigurace, složitější obsluha nástrojů a nemožnost abstrakce topologie. Tyto nedostatky jsou odstraněny aplikačním rozhraním popsaným v sekci 3.3.2. Nepřítomnost programu Dummynet v operačním systému Linux CentOS nebylo považováno za nevýhodu. Dummynet lze pro tento typ operačního systému zkompilovat jako externí modul jádra a poté nainstalovat, viz návod dostupný zde [42].

Pro úpravu datového toku jsou využívány tři funkce nástroje Traffic Control `qdisc`, `class` a `filter`. Řízení front `qdisc` již bylo částečně popsáno v kapitole 2.1.2. Jedná se o zásadní funkci, která obsluhuje jednotlivé typy front, včetně NetEm. Ukázka jednoduchého nastavení zpoždění 100 ms pro veškerý provoz na výstupu ze síťového rozhraní eth0 je na výpise 3.1. Tento příkaz přidá nový parametr řízení

Výpis 3.1: Ukázka jednoduchého nastavení zpoždění 100 ms.

```
1 # tc qdisc add dev eth0 root netem delay 100ms
```

fronty rozhraní eth0. V případě potřeby změnit hodnotu zpoždění není možné použít stejný příkaz, poněvadž je již parametr řízení fronty rozhraní eth0 nastaven. Není tedy možné přidat další, ale je nutné změnit stávající nastavení záměnou klíčového slova „add“ za „change“.

Sdílení nastavení jedné třídy systému třídou druhou je nezbytné pro nastavení více parametrů řízení front k jednomu typu datového toku. Vytvoření kořene stromu tříd a připojení konkrétního nastavení k dané větvi stromu je ukázáno na výpisu 3.2.

Výpis 3.2: Hiearchie tříd v nástroji Traffic Control.

```
# tc qdisc add dev eth0 handle 1: root htb
# tc class add dev eth0 parent 1: classid 1:1 htb rate 100Gbps
# tc qdisc add dev eth0 parent 1:1 handle 10: netem delay 100ms
# tc qdisc add dev eth0 parent 10:1 handle 101: corrupt 10P%
# tc class add dev eth0 parent 1: classid 1:2 htb rate 100Gbps
# tc qdisc add dev eth0 parent 1:2 handle 20: netem loss 20P%
```

Poslední funkcí Traffic Control využívané při emulaci je `filter`. Filtr je využíván pouze u hierarchického systému front kde rozhoduje o tom, která třída řízení front bude na daný provoz aplikována. Filtr nástroje TC má širokou paletu možných podmínek, dle kterých jsou pakety filtrovány. Jednou z možností filtrování je na základě značky „mark“ v záhlaví datové jednotky viz ukázka 3.3.

Výpis 3.3: Ukázka použití funkce filter nástroje Traffic Control.

```
# tc filter add dev eth0 protocol ip parent 1: handle 1 fw flowid
   1:1
```

Z ukázek příkazů v této části textu je patrné kaskádové skládání příkazových řetězců, kde úvodní část je vždy stejná, mění se pouze typ funkce. Této skutečnosti je využito při vytváření příkazů v obslužné části, jak bude detailněji popsáno níže.

Pro vytvoření značky lze užít jakýkoliv jiný nástroj, který může obsahovat vlastní podmínky značkování založené například na analýze datového provozu aplikační vrstvy. V případě emulátoru byl použit nástroj Iptables.

Iptables je stejně jako Traffic Control součástí jádra operačních systémů Linux a primárně slouží jako firewall. Nástroj Iptables nabízí větší možnosti filtrace než zmíněný filtr nástroje TC a pokrývá tři hlavní vrstvy RM ISO/OSI – linkovou, sítovou a transportní. Pro značkování datových jednotek je využívána tablka `mangle`.

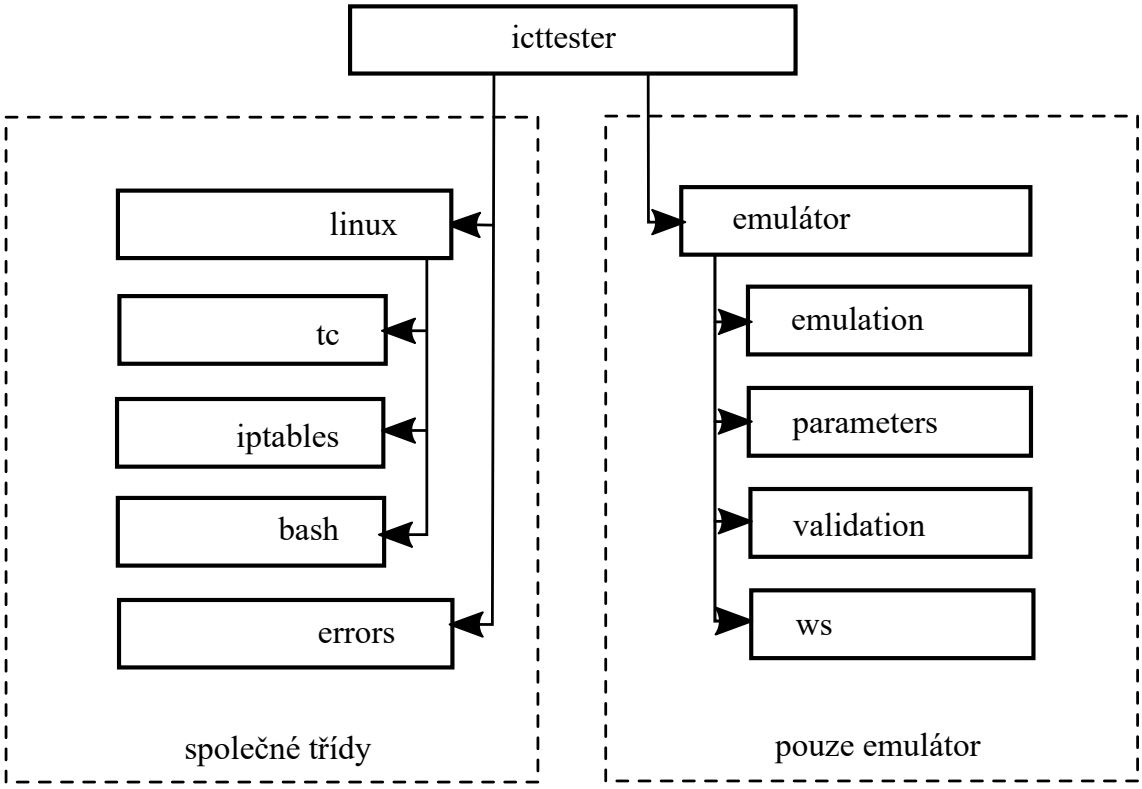
3.3.2 Webová služba

Jak bylo zmíněno výše, obslužná část emulátoru je realizována pomocí webové služby. Jako platforma pro vývoj webové služby byly použity technologie programovacího jazyka Java.

Celá webová služba je hostována na webovém serveru Apache Tomcat, což je volně dostupný software vyvíjený pod záštitou Apache Software Foundation. Je napsán v jazyce Java a poskytuje prostředí pro spouštění kódu tohoto programovacího jazyka. Více informací lze nalézt na oficiálních stránkách [43]. Pro hostování API emulátoru byla vybrána nejnovější verze 9, která již podporuje Java 8. Pro operační systém CentOS však 9. varianta serveru Tomcat není přítomna v balíčkovacím systému a musí být naistalována manuálně.

Pro usnadnění správy sestavení projektu je využíván nástroj Apache Maven. Stejně jako výše popsaný Tomcat je i Maven spravován komunitou Apache Software Foundation a je volně ke stažení. Maven je založen na principu popisování projektu pomocí POM (Project Object Model), který má strukturu XML a je uložen v hlavním adresáři projektu v souboru pom.xml. Informace jsou čerpány z domovské stánky Apache Maven [44]. Maven je v projektu emulátoru využíván především pro správu závislostí a balíčků frameworku Spring. Framework Spring je kontejner pro usnadnění vývoje Java EE (Enterprise Edition) aplikací včetně webové služby [45]. Ukázka správy kontejneru Spring v souboru pom.xml vyvinutého API je na výpise 3.4.

Podle funkcionality tříd programovacího jazyka Java lze obslužnou část rozdělit na část, která souvisí s operačním systémem a je u ní předpoklad, že bude využita i mimo projekt emulátoru a část, která souvisí pouze s emulátorem. Situace je naznačena na obrázku 3.9.



Obr. 3.9: Rozdělení obslužné části.

Výpis 3.4: Ukázka správy Spring v pom.xml.

```
<project ... >
  ...
  <properties>
    <java.version>1.8</java.version>
    <start-class>eu.gity.icttester.emulator.ws.Application</start-
    class>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    ...
  </dependencies>
   ...
</project>
```

Společné třídy

Jak bylo zmíněno výše (3.3.1), nástroj Traffic Control je ovládán příkazy přes příkazovou řádku operačního systému a generování těchto příkazů na základě požadavků uživatele je hlavním úkolem obslužné části.

K tvorbě příkazů nástroje Traffic Control slouží třídy balíčku `eu.gity.icttester.linux.tc`. Struktura příkazu se skládá z jednotlivých částí a na základě principu skládání příkazů byla rozdělena i logika tříd. Hlavní třídou zmíněného balíčku je třída `Tc`, která obsahuje metody pro přidání, změnu a smazání dané funkce nástroje Traffic Control, tj. skládá příkaz po argument „dev“, který je pro všechny tři funkce vždy stejný. O tvorbu příkazu se stará privátní metoda `createCommand`, jejímiž vstupními argumenty jsou typ operace, potomek, který metodu využije a doplňující řetězec příkazů, viz ukázka 3.5.

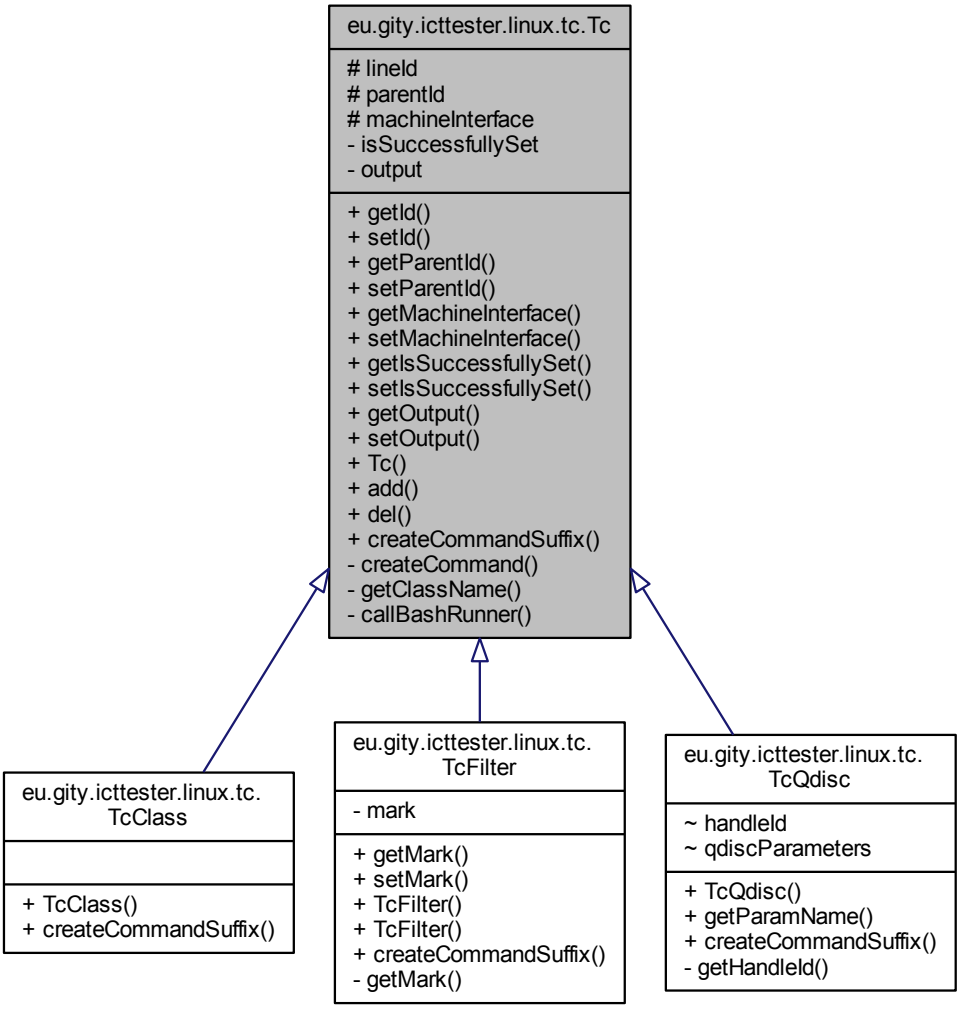
Výpis 3.5: Metoda createCommand třídy Tc.

```
1  private void createCommand(String operation, Tc caller, String
       commandStringSuffix) {
2          String commandString = "";
3          commandString += "tc ";
4          commandString += this.getClassName(caller) + " ";
5          commandString += operation + " ";
6          commandString += "dev " + this.machineInterface + " ";
7          if (!caller.getClass().getSimpleName().equals("TcQdisc")) {
8              commandString += "parent " + this.parentId + " ";
9          }
10         commandString += commandStringSuffix;
11
12         this.callBashRunner(commandString);
13 }
```

Každá z možností nástoje Traffic Control má svou vlastní třídu, která je potomkem třídy Tc. Třídy jsou pojmenovány shodně s funkcemi Traffic Control s prefixem Tc, tj. TcClass, TcFiler a TcQdisc. Diagram dědičnosti je na obrázku 3.10. Třída TcQdisc dále využívá funkcí třídy Netem, která vytváří části příkazů souvisejících s nástrojem NetEm. Takto rozvržená struktura tříd umožňuje snadné doplnění programu o další typy front či úpravu již implementovaných systémů.

Analogicky k třídám z balíčku `linux.tc` funguje třída Iptables z balíčku `linux.iptables`, která obsluhuje stejnojmenný nástroj operačního systému.

Třídy z balíčků `linux.tc` a `linux.iptables` nepokládají příkazy příkazové řádce operačního systému přímo, ovšem využívají metodu `runCommand` třídy BashRunner z balíčku `eu.gity.icttester.linux.bash`. Pro samotné provedení příkazu v příkazové řádce byla použita metoda `exec` ze třídy Runtime. Výstup z příkazové řádky je poté uložen do instance třídy BashOutput, která slouží jako kontejner pro výstupní řetězec a označení typu výstupu.

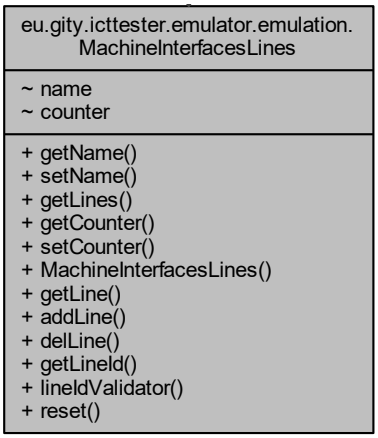


Obr. 3.10: Diagram dědičnosti pro třídu Tc.

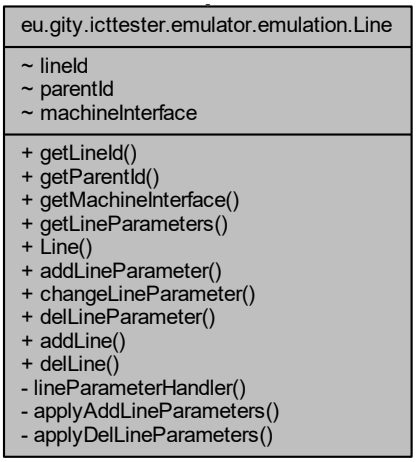
Třídy emulátoru

Druhou hlavní činností emulátoru je udržovat informace o jednotlivých nastavených parametrech. Tuto funkcionalitu obstarávají pole jednotlivých spojení a to pro každé síťové rozhraní zvlášť. Síťové rozhraní je zastupováno třídou MachineInterfacesLines, jenž obsahuje jméno rozhraní, pole s jednotlivými spojeními a jejich počítadlo. Dále poskytuje obslužné metody k těmto atributům. Spojení je charakterizováno třídou Line. Tato třída obsahuje všechny údaje pro realizaci spojení nástrojem Traffic Control, tj. identifikační číslo spojení, číslo jeho nadřazeného spojení, rozhraní, ke kterému je aplikováno a seznam emulovaných parametrů. Diagramy těchto tříd jsou

na obrázcích 3.11 a 3.12.



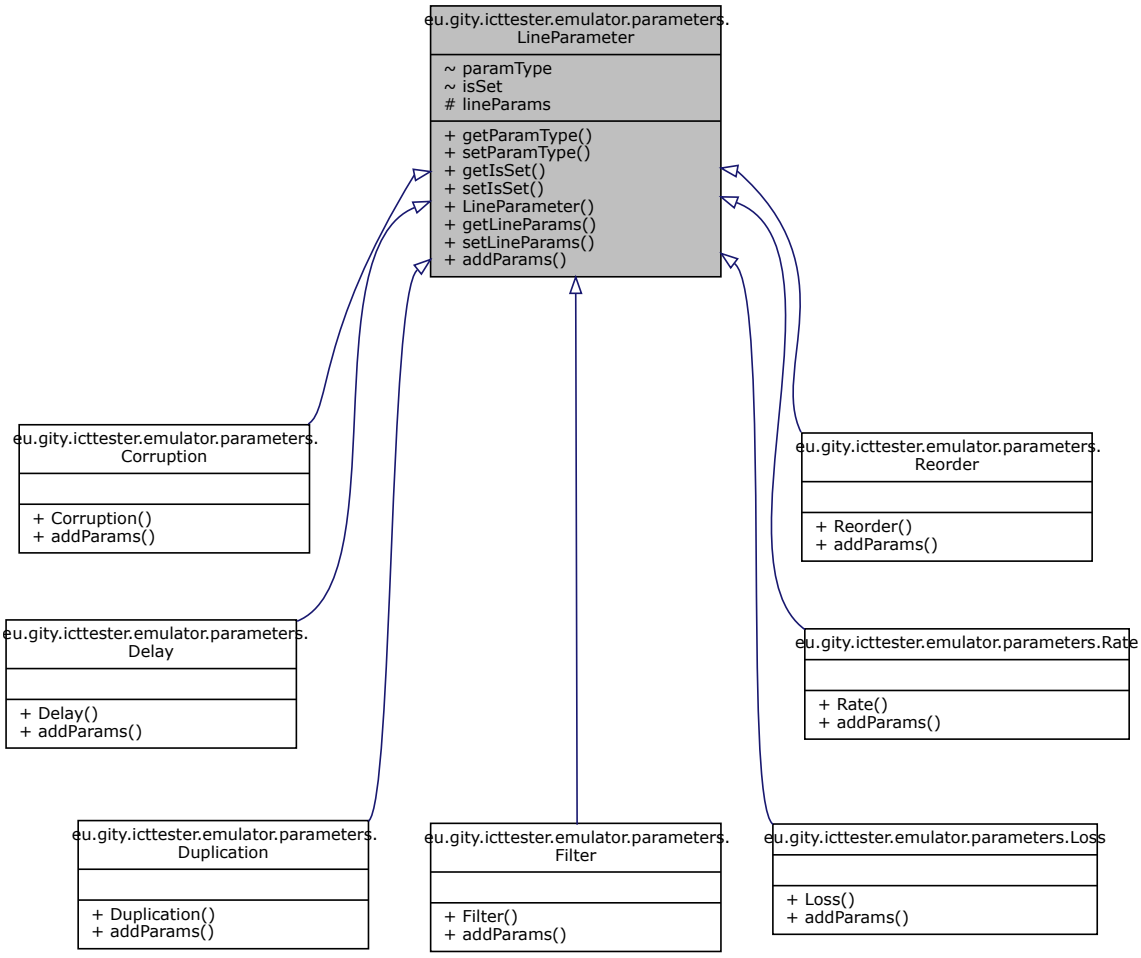
Obr. 3.11: Diagram třídy MachineInterfacesLines.



Obr. 3.12: Diagram třídy Line.

Přenosové parametry zmíněné výše jsou reprezentovány třídami v balíčku `emulator.parameter`. Třídy všech parametrů, tj. třídy Filter, Rate, Delay, Reorder, Duplication, Loss a Corruption, jsou odvozeny od abstraktní rodičovské třídy LineParameter. Celá struktura je zobrazena na diagramu tříd na obrázku 3.13.

Pro validaci vstupních dat byly implementovány statické metody obsažené ve třídě Validation. Pro ověření správnosti vstupního řetězce je využito regulárních výrazů. Ukázka metody pro ověření IP adresy je vypsána zde 3.6.



Obr. 3.13: Diagram dědičnosti třídy LineParameter.

Výpis 3.6: Metoda createCommand třídy Tc.

```
public static ErrorHandler ipv4Address(String stringToValid) {
        if (Pattern.matches(
                "^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
    \\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
    \\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\$",
                stringToValid)) {
            return new ErrorHandler(0);
        } else {
            String errorString = "Value is not IP address v4.";
            return new ErrorHandler(1, errorString);
        }
}
```

Třetí nezbytnou součástí webového API je samotný kontrolér webové služby, tj. metody, které jsou volány klientem. Tyto metody se nacházejí ve třídě MainController balíčku `emulator.ws.controllers`. Poskytované metody a jejich popis jsou vypsány v tabulkách 3.1 a 3.2.

Tab. 3.1: Metody webové služby – část A.

Název metody	Popis metody
init	Metoda, která inicializuje celý emulátor.
addLine	Metoda, která přidá novou linku pro dané rozhraní stroje.
getLine	Metoda, která vrátí danou linku daného rozhraní stroje.
getLines	Metoda, která vrátí všechny linky daného rozhraní stroje.
getAllLines	Metoda, která vrátí všechny linky pro všechna rozhraní stroje.
delLine	Metoda, která smaže danou linku daného rozhraní stroje.
addFilter	Metoda, která přidá dané lince daného rozhraní stroje parametr filtr.
changeFilter	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru filtr.
delFilter	Metoda, která odstraní dané lince daného rozhraní stroje parametr filtr.
addRate	Metoda, která přidá dané lince daného rozhraní stroje parametr propustnost.
changeRate	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru propustnost.
delRate	Metoda, která odstraní dané lince daného rozhraní stroje parametr propustnost.
addDelay	Metoda, která přidá dané lince daného rozhraní stroje parametr zpoždění.

Tab. 3.2: Metody webové služby – část B.

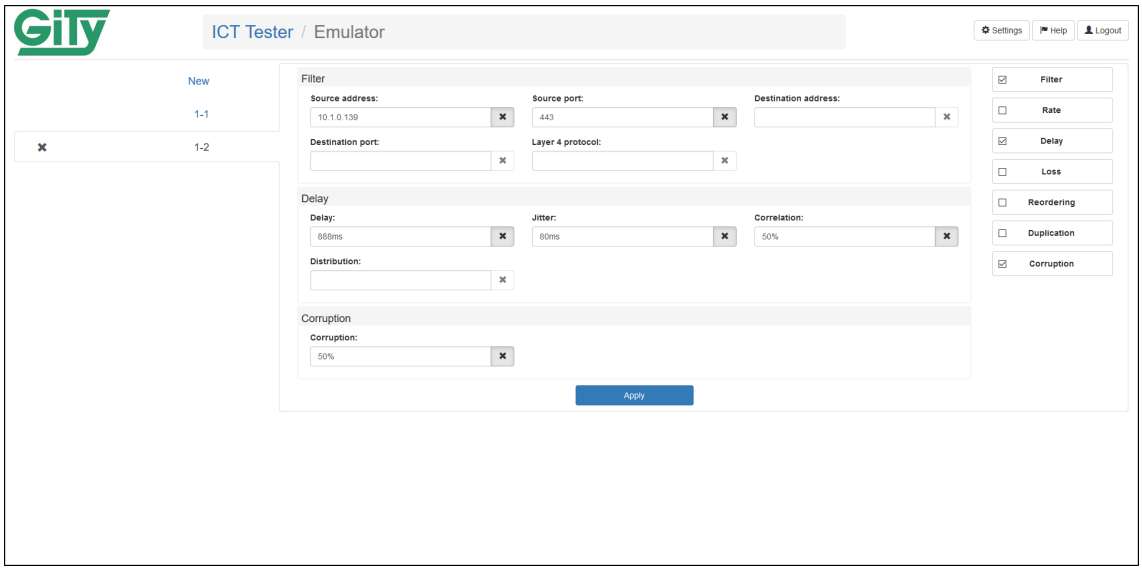
Název metody	Popis metody
changeDelay	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru zpoždění.
delDelay	Metoda, která odstraní dané lince daného rozhraní stroje parametr zpoždění.
addReorder	Metoda, která přidá dané lince daného rozhraní stroje parametr záměna pořadí.
changeReorder	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru záměna pořadí.
delReorder	Metoda, která odstraní dané lince daného rozhraní stroje parametr záměna pořadí.
addDuplication	Metoda, která přidá dané lince daného rozhraní stroje parametr duplikace.
changeDuplication	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru duplikace.
delDuplication	Metoda, která odstraní dané lince daného rozhraní stroje parametr duplikace.
addLoss	Metoda, která přidá dané lince daného rozhraní stroje parametr ztrátovost.
changeLoss	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru ztrátovost.
delLoss	Metoda, která odstraní dané lince daného rozhraní stroje parametr ztrátovost.
addCorruption	Metoda, která přidá dané lince daného rozhraní stroje parametr chybovost.
changeCorruption	Metoda, která změní dané lince daného rozhraní stroje parametr(y) parametru chybovost.
delCorruption	Metoda, která odstraní dané lince daného rozhraní stroje parametr chybovost.

3.3.3 Webové uživatelské rozhraní

K vývoji webového uživatelského rozhraní byly použity známé technologie HTML, CSS a JavaScript. Jazyk HTML je použit pouze pro vytvoření hlavní struktury webové stránky, většina obsahu je generována a dynamicky měněna metodami vytvořenými v technologii JavaScript. Kaskádové styly pak definují grafický styl vygenerovaných komponent stránky. Pomocným nástrojem při výstavbě webových stránek byl nástroj Bootstrap, který zjednodušuje tvorbu responsivního webu a obsahuje mnoho užitečných komponent jako jsou tlačítka, navigační lišty nebo formuláře.

Vytvořené uživatelské rozhraní nevyužívá žádné serverové technologie a je tak plně klientskou aplikací. Pro ovládání emulátoru a získávání informací o nastavení emulátoru je využíváno webové služby popsané v podkapitole 3.3.2.

Při prvním načtení stránky se automaticky načtou všechna konfigurovaná spojení emulátoru zavoláním metody webové služby `getAllLines`. Na základě takto získaných dat je poté funkcí `mainCreator` vytvořen obsah stránky. Při generování obsahu je využíváno konfigurační proměnné `config`, která obsahuje všechny nastavitelné parametry emulátoru. Pokud by byl v budoucnu požadavek na rozšíření aplikace o další parametr, postačí změnit tuto konfigurační předlohu. Cykly vykreslující vzhled stránky novou komponentu automaticky použijí. Ukázka vzhledu se dvěma konfigurovanými spoji je na obrázku 3.14.

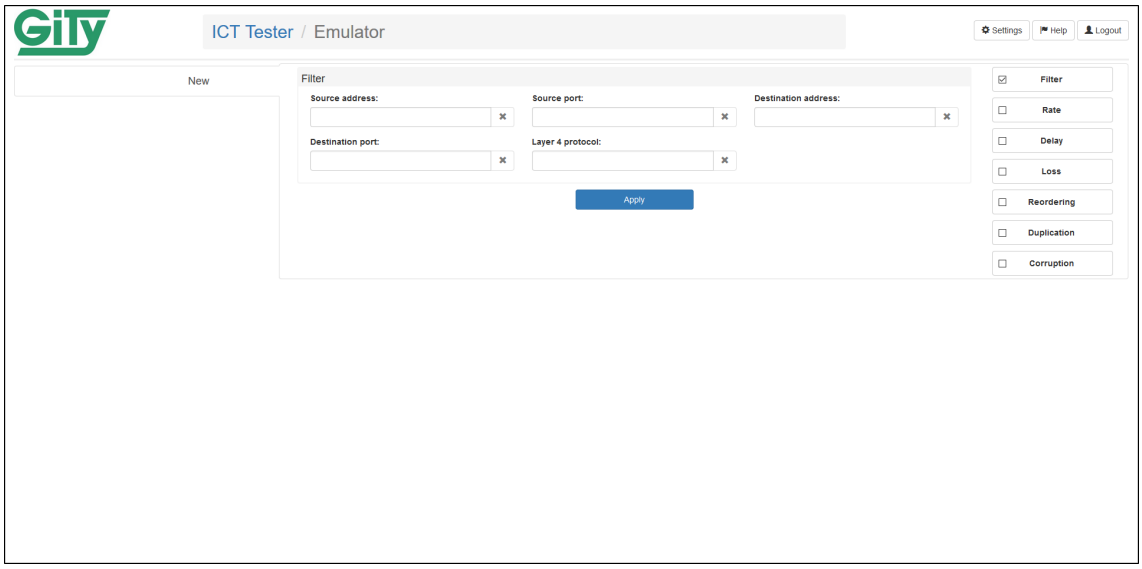


Obr. 3.14: Ukázka WUI se dvěma konfigurovanými spoji.

V levé části je menu, ze kterého lze vybírat jednotlivé již konfigurované spoje. Hlavní část stránky funguje na principu přepínání záložek. Při kliknutí na jinou položku menu – záložku – se zavolá funkce `menuButtonClick`, která překreslí obsah

stránky dle zvoleného spojení. Obsahem záložky jsou nastavení jednotlivých parametrů daného spojení. Pro větší přehlednost lze formulář daného parametru skrýt pomocí příslušného tlačítka na pravé straně obrazovky. Je-li daný parametr nastaven, je formulář daného parametru automaticky viditelný, ostatní jsou defaultně skryty. Tato funkcionalita je obsluhována funkcí `lineCheckboxesHandler`.

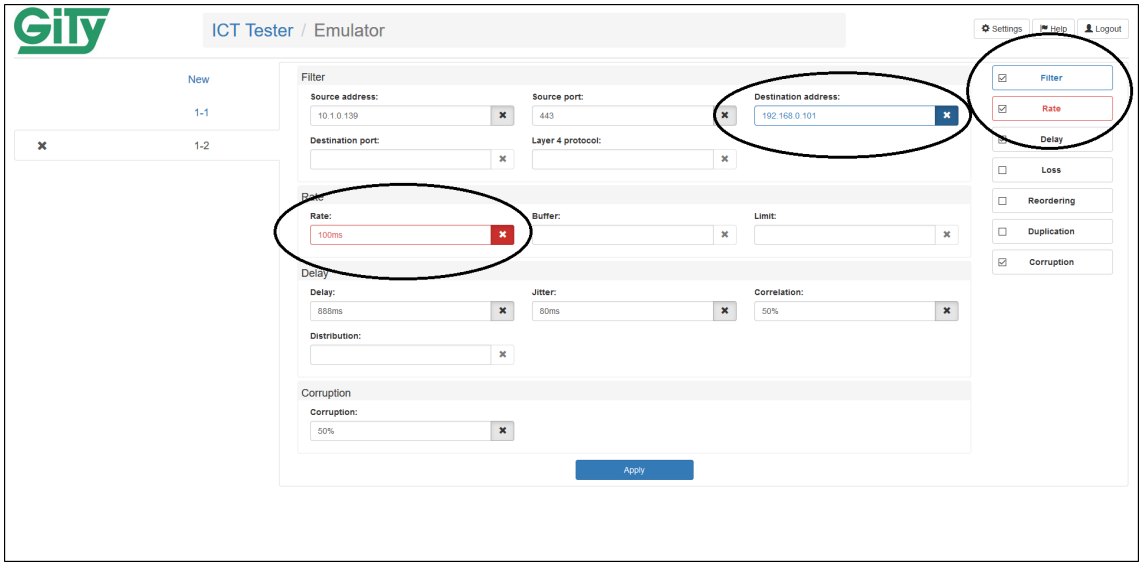
Není-li z webové služby získáno žádné existující spojení, je nabízeno pouze vytvoření nového spojení, viz ukázka na obrázku 3.15.



Obr. 3.15: Ukázka WUI s žádným konfigurovaným spojem.

Přidání anebo změna hodnoty parametru je stejně jako u webové služby validována. Pro validaci je využívána funkce `validator`, která pro ověřování správnosti formátu vstupních dat používá regulární výrazy. Jsou-li vložená data správná, obarví se okraje vstupního okna modrou barvou na znamení změněné hodnoty a stejně tak se zbarví i tlačítko pro schování formuláře parametru v pravé části stránky. Neprojde-li vstupní hodnota kontrolou, je vyvolána výstražná hláška a textové okno je orámováno červenou barvou. Stejně je zbarveno i tlačítko pro schování formuláře parametru. O grafické znázornění výsledku validace se stará funkce `formInputParamHandler`. Ukázka výstupu validity dat je na obrázku 3.16.

Je-li vložena hodnota správná, je kromě výše popsané grafické změny provedeno její vložení do proměnného pole `parametersArray` objektu `changesClass`. Při stiskunít tlačítka „Apply“ jsou změny v cyklu procházeny a generovány URL dotazy na webovou službu. O volání webové služby a zpracování výsledků se starají funkce `createLineHttpRequest`, `changeLineHttpRequest`, `deleteLineHttpRequest`.



Obr. 3.16: Ukázka validace WUI.

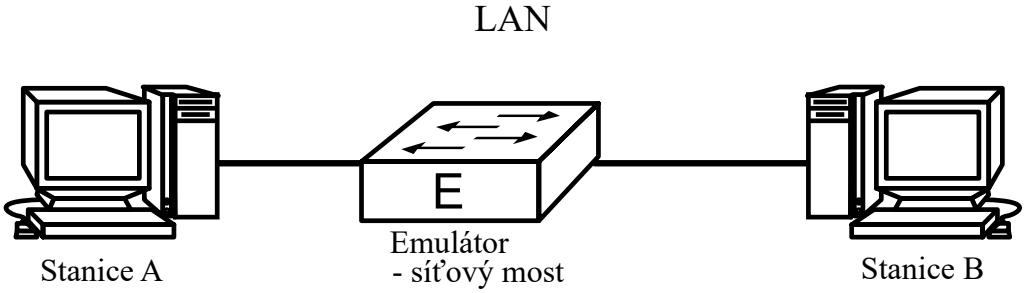
4 MĚŘENÍ ŘEŠENÉHO EMULÁTORU

V této kapitole je věnována pozornost měření schopností emulátoru měnit hodnoty přenosových parametrů v datových sítích. Toto měření bylo uskutečněno v rámci testování vyvíjeného přístroje jakožto ověření funkčnosti emulovat klíčové přenosové parametry:

- propustnost – podsekce 4.2.1,
- zpoždění a jitter – podsekce 4.2.2,
- záměnu pořadí – podsekce 4.2.3,
- duplikaci – podsekce 4.2.4,
- ztrátovost – podsekce 4.2.5.

4.1 Popis měření

Měření bylo prováděno v laboratoři na testovací soustavě, jejíž topologie je zobrazena na obrázku 4.1. Z obrázku je patrné, že emulátor byl nastaven do módu síťového mostu mezi koncovými stanicemi Stanice A a Stanice B. Všechny prvky tak náležely jedné lokální síti bez možnosti přístupu do jiné sítě, tj. v plně autonomním prostředí. Jedná se tedy o scénář popsaný v kapitole 3.2 podsekci 3.2.1. Pro usnadnění nastavení testovací sítě byl využit DHCP server implementovaný v emulátoru. Podrobný popis všech prvků v topologii následuje.



Obr. 4.1: Topologie testovací soustavy.

4.1.1 Emulátor

Softwarová část emulátoru byla nasazena na hardware serverové stanice Hewlett-Packard HP ProLiant ML110. Jednalo se o provizorní řešení pouze pro vývoj a testování vyvíjeného softwaru. Moderní a výkonný hardware, optimalizovaný na míru vyvíjeného ICT testeru, nebyl v době, kdy byla psána tato práce, k dispozici. Hardwarová konfigurace emulátoru je vypsána v tabulce 4.1.

Tab. 4.1: Hardwarová konfigurace emulátoru.

Procesor	Intel(R) Xeon(R) CPU 3040 @ 1,86 GHz
Počet jader	2
Paměť RAM	4,00 MB
Síťové rozhraní 1	HP PCIe Gigabit Server
Síťové rozhraní 2	Intel PRO/1000 PT Desktop

4.1.2 Stanice A

Koncový síťový uzel Stanice A byl realizován přenosným počítačem Samsung N150, jehož hardwarová konfigurace je vypsána v tabulce 4.2. Operačním systémem byl Linux Mint 17.1 MATE 32-bit.

Tab. 4.2: Hardwarová konfigurace Stanice A 1.

Procesor	Intel Atom N450 (1.66 GHz, 512 KB Cache)
Počet jader	2
Paměť RAM	1,00 MB
Síťové rozhraní	Marvell 88E8040 PCI-E Fast Ethernet Controller

Při testování propustnosti byl uzel Stanice A zastoupen přenosným počítačem Acer Aspire 5750G, jehož hardwarová konfigurace je vypsána v tabulce 4.3. Operačním systémem byl Microsoft Windows 10.1 Pro.

Tab. 4.3: Hardwarová konfigurace Stanice A 2.

Procesor	Intel(R) Core(TM) i5-2410M CPU @ 2,30 GHz
Počet jader	4
Paměť RAM	8,00 MB
Síťové rozhraní	Broadcom NetLink (TM) Gigabit Ethernet

4.1.3 Stanice B

Koncový síťový uzel Stanice B byl realizován přenosným počítačem Dell Latitude E6420, jehož hardwarová konfigurace je vypsána v tabulce 4.4. Operačním systémem byl Linux Debian 8: 8.4 32-bit.

Tab. 4.4: Hardwarová konfigurace Stanice B.

Procesor	Intel(R) Core(TM) i3-2520M CPU @ 2,50 GHz
Počet jader	4
Paměť RAM	6 MB
Síťové rozhraní	Intel(R) 82579LM Gigabit Network Connection

4.1.4 Kabeláž

Pro propojení jednotlivých uzlů v testovací síti byly využity běžně dostupné ethernetové kabely UTP kategorie Cat5e s délkou 2,5 m.

4.1.5 Program Iperf3

Program Iperf3 je nástroj vyvíjený za účelem měření propustnosti, konstantního i proměnného zpoždení a dalších přenosových parametrů internetových sítí [49]. Program funguje na principu klient-server, kde klient generuje TCP anebo UDP pakety, které posílá na serverovou stanici. Nástroj byl původně vyvíjen pro operační systémy Linux, nyní však již existují verze také pro Windows. Program Iperf3 lze ovládat pomocí příkazové řádky anebo lze nástroj získat integrovaný v grafických aplikacích, které kromě nastavení nástroje dokáží poskytovat uživateli zpětnou vazbu pomocí vykreslovaných grafů. Ukázka výstupu programu Iperf3 v příkazové řádce je na výpise 4.1. Program Iperf3 byl využit při měření propustnosti a všech typů zpoždení.

4.1.6 Program Ping

Měření parametrů duplikace a ztrátovost bylo provedeno známým nástrojem Ping, využívajícím ICMP pakety. Program Ping vyšle ICMP paket k cíli a měří, za jak dlouhou dobu se na odeslaný paket vrátí z cíle odpověď, měří tedy obousměrné zpoždění (viz 1.2.2). Při emulaci se ovšem emuluje pouze jednosměrné zpoždění, a proto pro měření zpoždění a jitteru není tento nástroj vhodný.

Kromě měření zpoždění dokáže nástroj Ping sledovat, zdali daný paket dorazil do cílového uzlu a také, zdali po cestě nebyl duplikován. Stejně jako zpoždění jsou i tyto parametry měřeny jak pro cestu od zdroje k cíli, tak pro cestu nazpět, od cíle ke zdroji. Vzhledem k velmi jednoduché topologii měření, která obsahuje pouze dva koncové uzly a emulátor, byla spolehlivost tohoto zpoždění 100 %. U parametrů duplikace a ztrátovost lze tak cestu zpět zanedbat a program Ping pro měření zmíněných parametrů použít.

Výpis 4.1: Ukázka výstupu programu Iperf3.

```
Connecting to host 192.168.88.102, port 5201
[  5] local 192.168.88.101 port 63938 connected to 192.168.88.102
   port 5201
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.01   sec  17.4 MBytes   144 Mbits/sec
[  5]   1.01-2.01   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   2.01-3.01   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   3.01-4.01   sec  11.5 MBytes  96,6 Mbits/sec
[  5]   4.01-5.01   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   5.01-6.01   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   6.01-7.00   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   7.00-8.00   sec  11.5 MBytes  96,6 Mbits/sec
[  5]   8.00-9.00   sec  11.4 MBytes  95,6 Mbits/sec
[  5]   9.00-10.02  sec  11.6 MBytes  96,2 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-100.01 sec  1.12 GBytes  96.3 Mbits/sec     sender
[  5]   0.00-100.01 sec  1.12 GBytes  96.3 Mbits/sec     receiver

iperf Done.
```

4.1.7 Nástroj pro měření záměny pořadí

Program Ping však není možné použít v případě měření záměny pořadí, poněvadž pracuje v iterativním režimu, tj. další paket je odeslán až po doručení předchozího či vypršení časovače. Pro měření záměny pořadí paketů byl tedy sestaven jednoduchý klient-server program využívající nástroje netcat pro posílání dat přes datovou síť.

Testovací program na straně klienta generoval UDP pakety, v jejichž těle byla jednoduchá číselná hodnota 1–1000000 označující pořadí vyslaného paketu. Na straně klienta byly všechny pakety zachyceny programem tcpdump a uloženy do souboru s příponou pcap. Nástrojem tshark poté byly z uloženého záznamu paketů automaticky získány hodnoty v datovém poli každého paketu a uloženy do textového souboru. Z tohoto souboru již pak bylo možné zjistit, zdali číslo v datovém poli paketu odpovídá číselné hodnotě pozice, na které byl paket přijat serverem. Ukázky skriptů a příkazů jsou v příloze B.

4.1.8 Výpočet ochylek

Při zpracování naměřených výsledků byly použity ukazatele absolutní a relativní odchylky, které jsou dále popsány. Definice těchto veličin jsou převzaty z [46], [47] a [48].

Absolutní odchylkou δ_{a} je rozuměn rozdíl mezi naměřenou hodnotou x_{m} a skutečnou, v případě emulace požadovanou, hodnotou x:

$$\delta_{\mathrm{a}} = x_{\mathrm{m}} - x. \tag{4.1}$$

Relativní odchylka δ_{r} je pak definována jako podíl absolutní chyby δ_{a} a požadované hodnoty x. Pro získání procentuální hodnoty se vztah násobí 100. Výpočet relativní odchylky je

$$\delta_{\mathrm{r}} = \frac{\delta_{\mathrm{r}}}{x} * 100 \qquad [\%]. \tag{4.2}$$

Pro větší názornost je relativní odchylka uváděna v absolutní hodnotě.

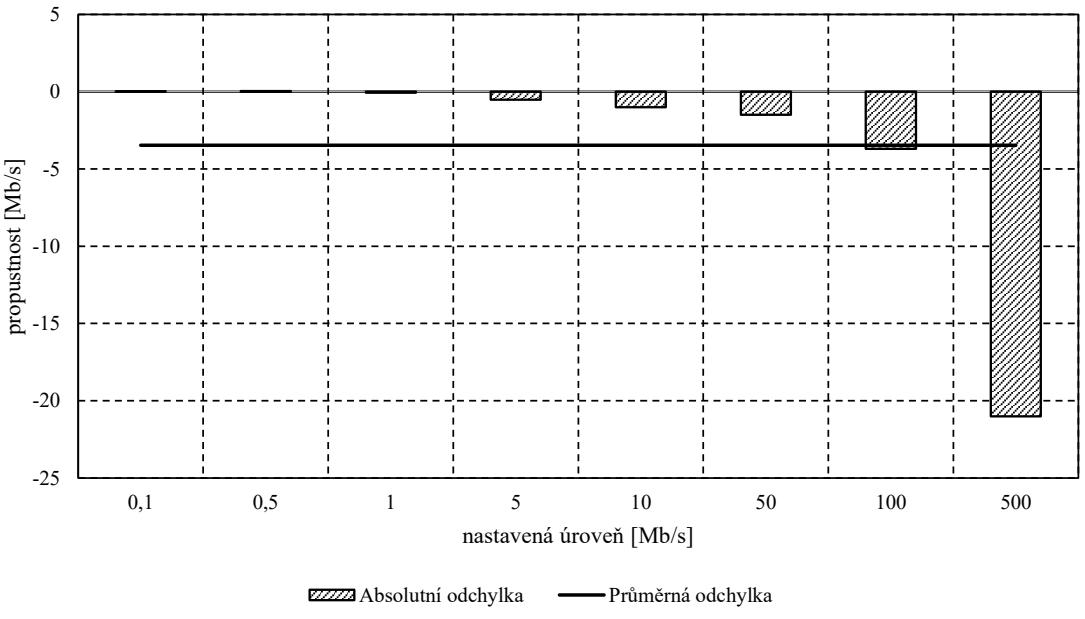
4.2 Měření klíčových přenosových parametrů

4.2.1 Propustnost

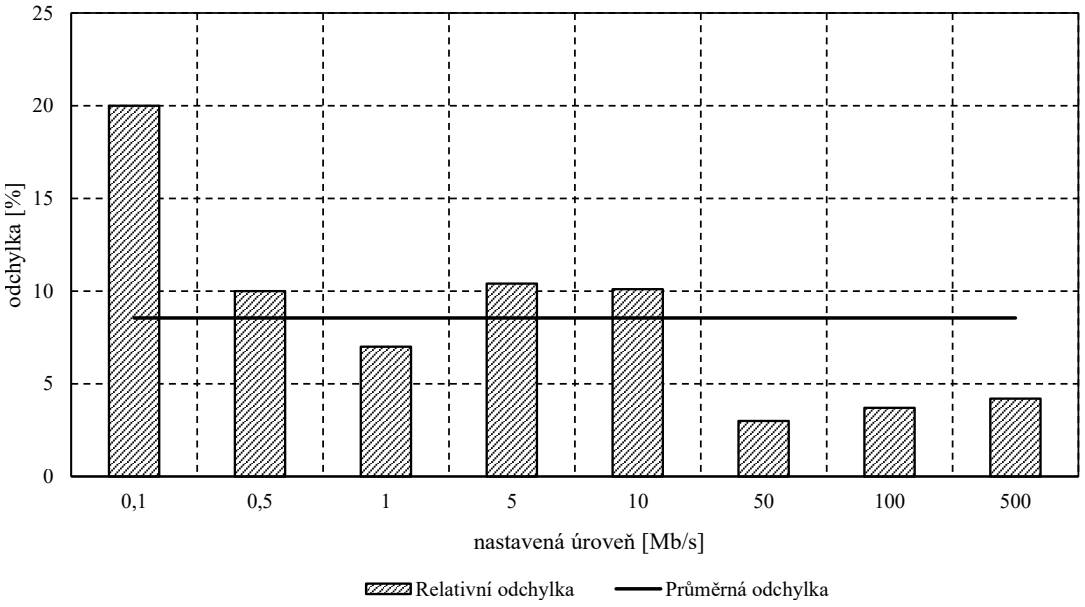
Měření emulace snižování propustnosti bylo provedeno pro osm vybraných úrovní tak, aby pokrývaly všechny zajímavé rychlosti, viz první sloupec tabulky 4.5. Pro každou z úrovní byl vyslán 1 milion paketů a výsledná hodnota propustnosti je jejich průměrem. Teoretická propustnost testovacího pracoviště byla 1 Gb/s, ovšem reálná naměřená maximální hodnota byla 824 Mb/s. V tabulce 4.5 jsou v prvním sloupci hodnoty požadované od emulátoru, ve druhém sloupci skutečně naměřené hodnoty a ve třetím a čtvrtém sloupci absolutní, resp. relativní, odchylka naměřených výsledků od požadovaných. Obě odchylky pro všechny hodnoty jsou vykresleny v grafech 4.2 a 4.3. Z grafu relativní odchylky vyplývá, že průměrná absolutní relativní odchylka je 8,55 Mb/s a pro vývoj odchylky nelze určit jasný trend. Přesnost emulace propustnosti je závislá na vhodném nastavení systému fronty a na výkonnosti hardwaru. Předpoklad, že kvalita emulace bude s rostoucí rychlostí přesnější, se však naplnil. Nejvyšší hodnoty dosáhla odchylka u požadované propustnosti 0,1 Mb/s. Velikost této odchylky je však umocněna velmi nízkou emulovanou propustností, absolutní odchylka pro zvolenou propustnost byla jen 0,02 Mb/s. Naopak u emulované propustnosti 500 Mb/s byla vypočtena relativní odchylka 4,2 %, což ovšem dělá rozdíl −21 Mb/s.

Tab. 4.5: Výsledky měření propustnosti.

Požadovaná úroveň [Mb/s]	Naměřená úroveň [Mb/s]	Absolutní odchylka [Mb/s]	Relativní odchylka [%]
0,1	0,12	0,02	20,00
0,5	0,45	-0,05	10,00
1,0	0,93	-0,07	7,00
5,0	4,48	-0,52	10,40
10,0	8,99	-1,01	10,10
50,0	48,50	-1,50	3,00
100,0	96,30	-3,70	3,70
500,0	479,00	-21,00	4,20



Obr. 4.2: Propustnost – absolutní odchylka.



Obr. 4.3: Propustnost – relativní odchylka.

4.2.2 Zpoždění a jitter

Měření schopnosti emulovat konstantní zpoždění bylo měřeno u osmi nastavených hodnot mezi 0–1000 milisekund tak, že pro každou úroveň zpoždění byl vyslán 1 milion paketů. Výsledná hodnota dané úrovně je poté získána jako aritmetický průměr všech zpoždění pro danou úroveň. Výsledky měření jsou vypsány v tabulce 4.6. V prvním sloupci tabulky jsou hodnoty požadované od emulátoru, ve druhém sloupci skutečně naměřené hodnoty a ve třetím a čtvrtém sloupci absolutní, resp. relativní, odchylka požadovaných a naměřených výsledků. Obě odchylky pro všechny hodnoty jsou vykresleny v grafech 4.4 a 4.5.

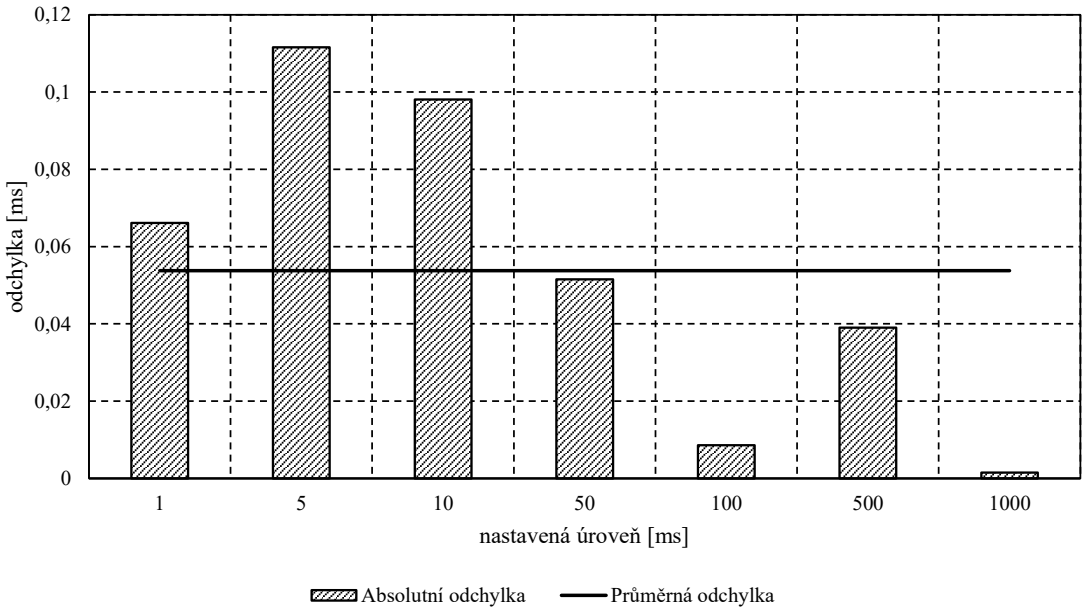
Z grafu relativní odchylky vyplývá, že relativní odchylka se s rostoucím nastaveným zpožděním snižuje téměř k nule. (Skutečná hodnota odchylky zpoždění 1000 ms je 0,00015 %.) Nejvyšší hodnoty 6,6156 ms dosahuje pro úroveň 1 ms, což ovšem v absolutní míře znamená pouhých 66,1560 μs. Celková průměrná absolutní odchylka je 0,0538 ms a relativní 1,4211 %.

Tab. 4.6: Výsledky měření konstantního zpoždění.

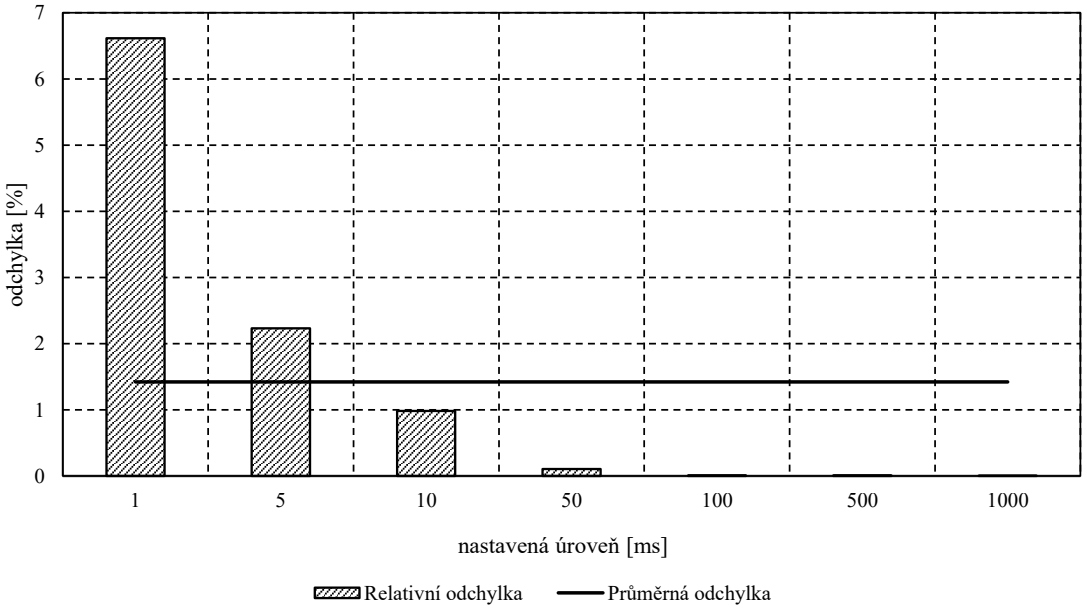
Požadovaná úroveň [ms]	Naměřená úroveň [ms]	Absolutní odchylka [ms]	Relativní odchylka [%]
0,00	0,13	–	–
1,00	1,07	0,07	6,62
5,00	5,11	0,11	2,23
10,00	10,10	0,10	0,98
50,00	50,50	0,05	0,10
100,00	100,01	0,01	0,01
500,00	500,04	0,04	0,01
1000,00	1000,00	0,00	0,00

Proměnné zpoždění, jitter, bylo měřeno při nastaveném zpoždění 100 ms s odchylkou ±20 ms pro 10 tisíc paketů. Výsledná střední hodnota proměnného zpoždění byla 100,0688 ms. Jednotlivé body měření jsou vyobrazeny v grafu 4.6. Z grafu rozložení 4.7 je dále patrné, že rozložení hodnot je relativně konstantní, tj. volba zpoždění daného paketu byla absolutně náhodná.

Proměnné zpoždění s normálním rozložením bylo měřeno se střední hodnotou 100 ms. Výsledky měření jsou uvedeny v grafech 4.8 a 4.9. Na grafu bodů zpoždění jednotlivých paketů, první zmiňovaný graf, je patrná vysoká hustota bodů v úrovni nastavené střední hodnoty zpoždění (100 ms), která se směrem od této hodnoty snižuje. Tato skutečnost je ještě lépe pozorovatelná z histogramu na obrázku 4.9, který vyjadřuje zastoupení jednotlivých naměřených hodnot pro daný interval zpoždění.



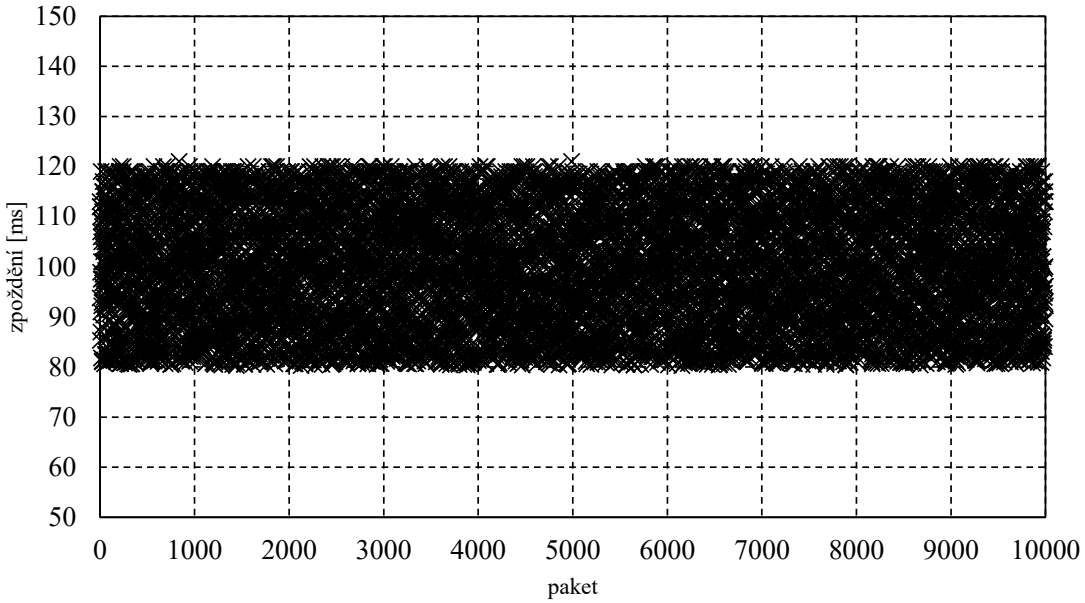
Obr. 4.4: Konstantní zpoždění – absolutní odchylka.



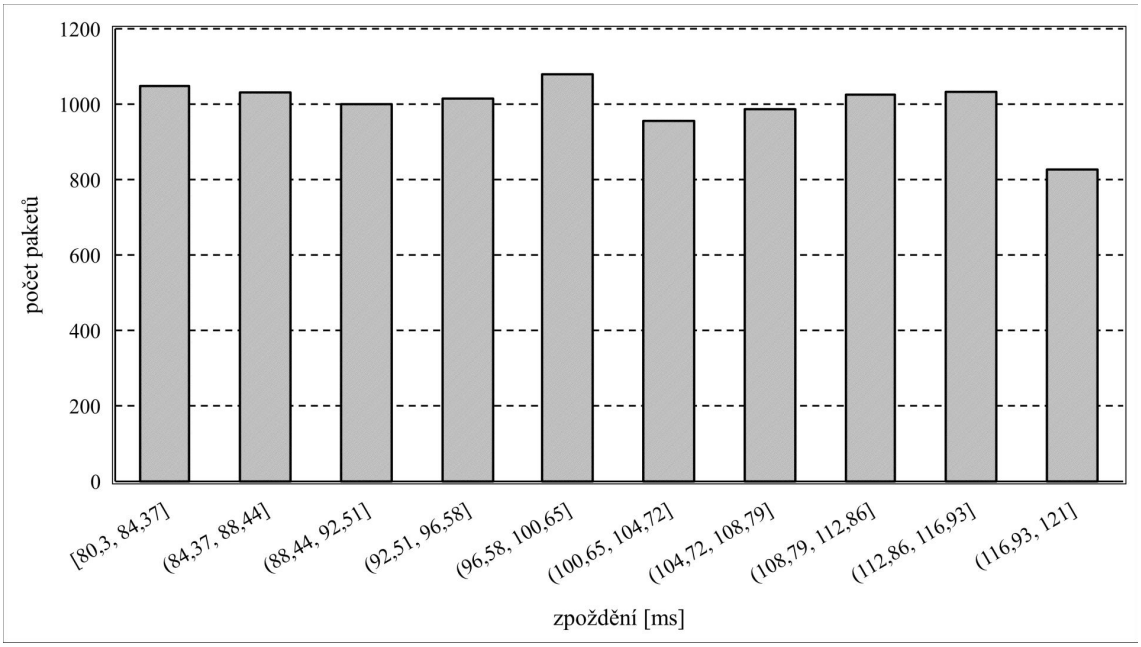
Obr. 4.5: Konstantní zpoždění – relativní odchylka.

Na základě výsledků vykreslených v grafech lze tedy potvrdit, že rozložení naměřených hodnot odpovídá normálnímu, Gaussovu, rozložení. Viz teorie v kapitole 1, konkrétně obrázek 1.4.

Dále bylo měřeno proměnné zpoždění s pareto a pareto-normálním rozložením.

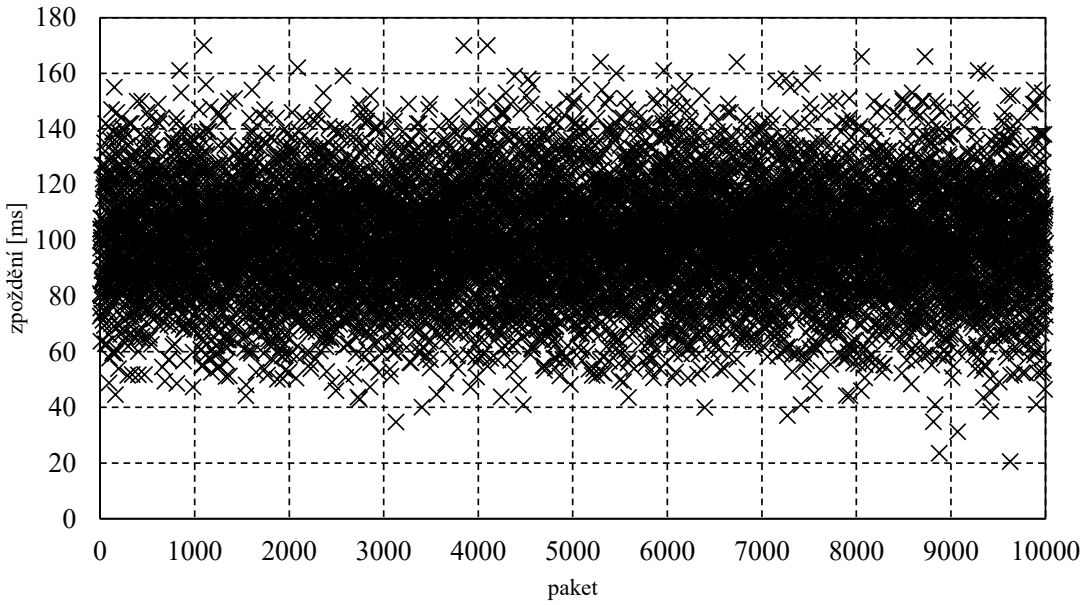


Obr. 4.6: Jitter 100 ±20 ms – naměřené hodnoty.

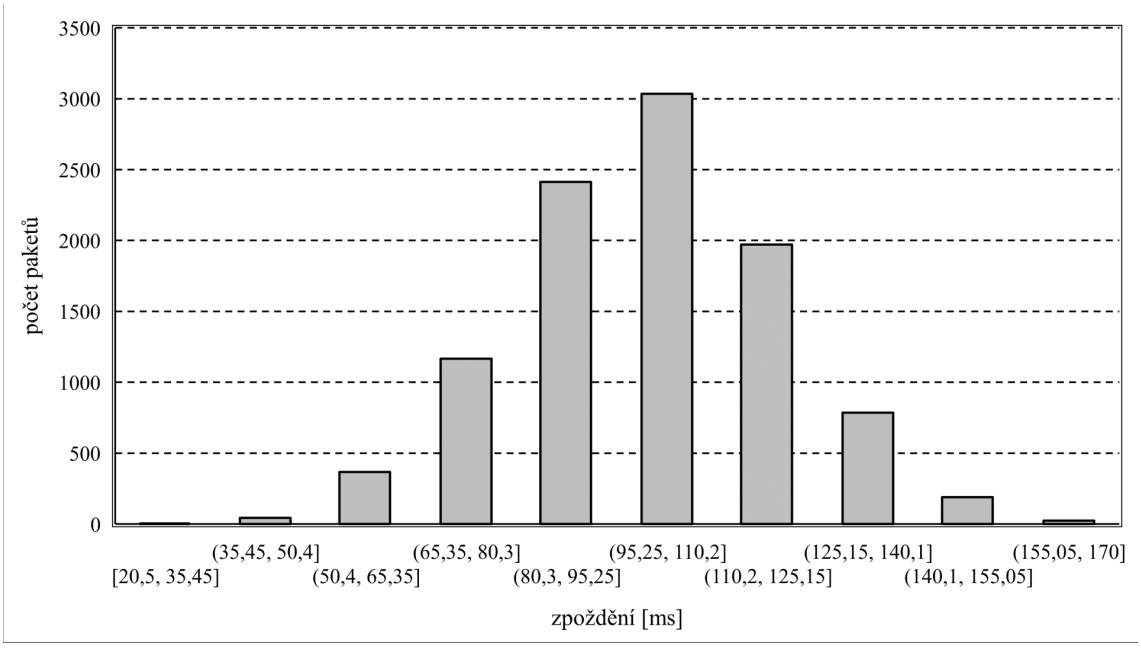


Obr. 4.7: Jitter 100 ±20 ms – histogram.

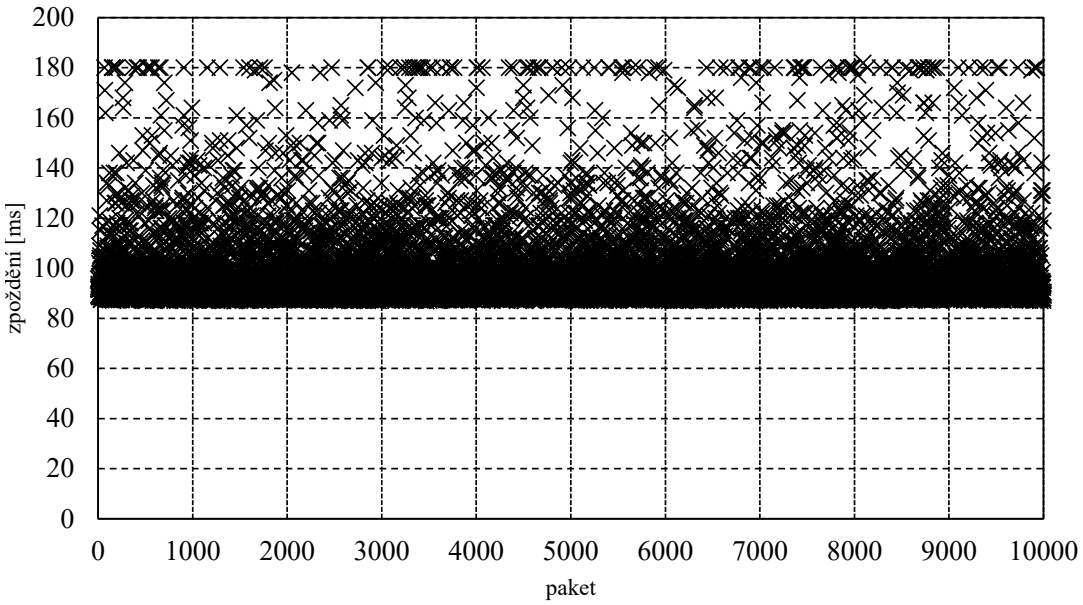
Výsledky obou měření také odpovídají teoretickým předpokladům a jsou vykresleny na obrázcích 4.10, 4.11 a 4.12, 4.13.



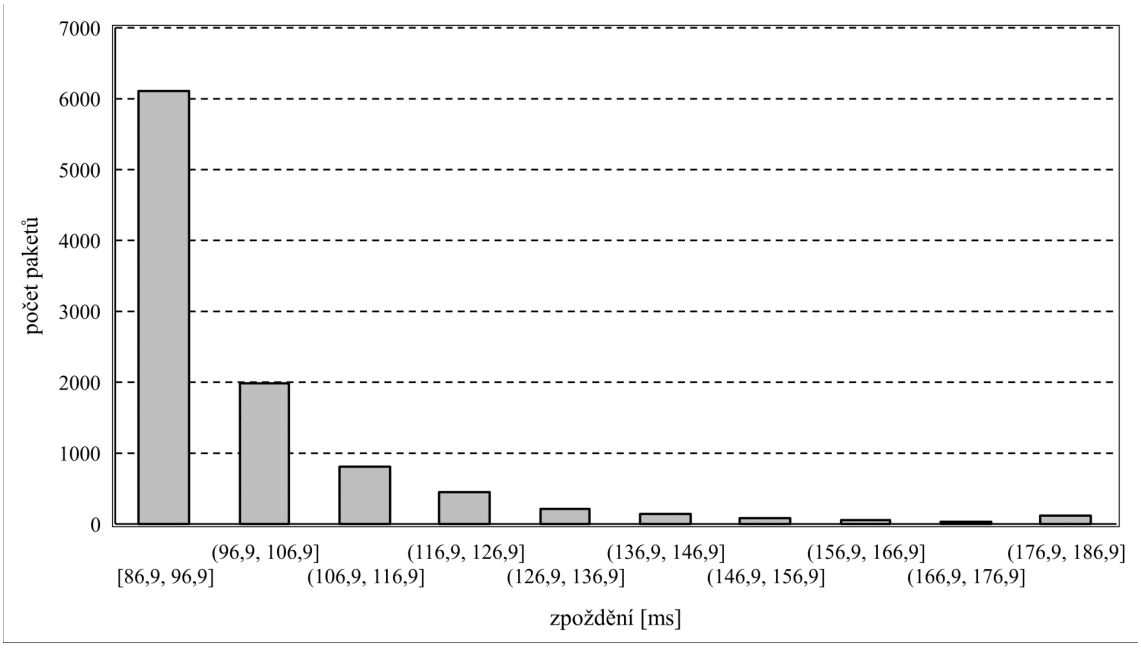
Obr. 4.8: Normální rozložení, 100 ms – naměřené hodnoty.



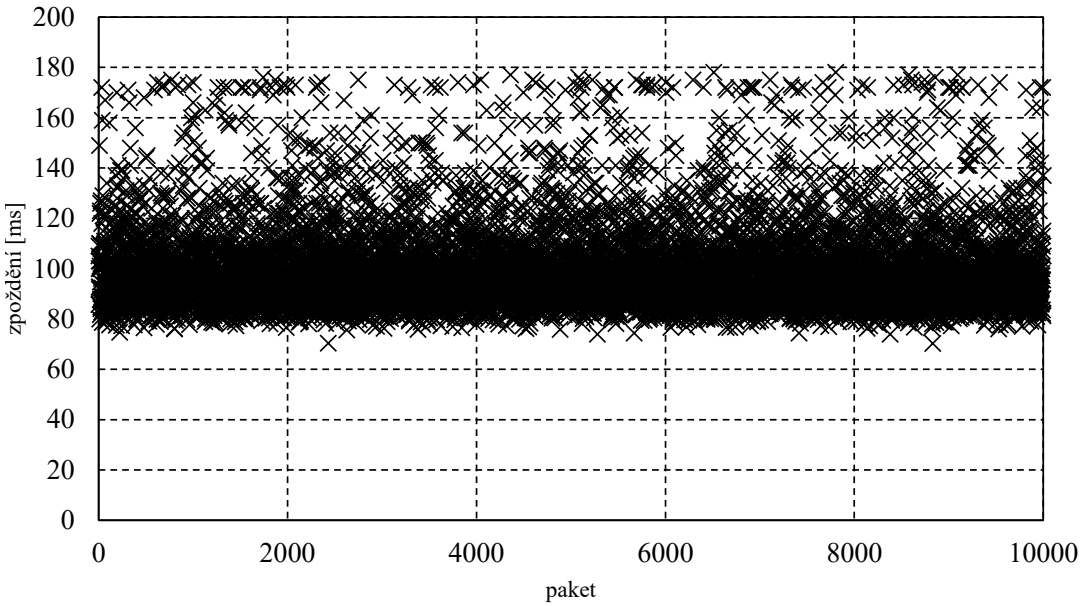
Obr. 4.9: Normální rozložení, 100 ms – histogram.



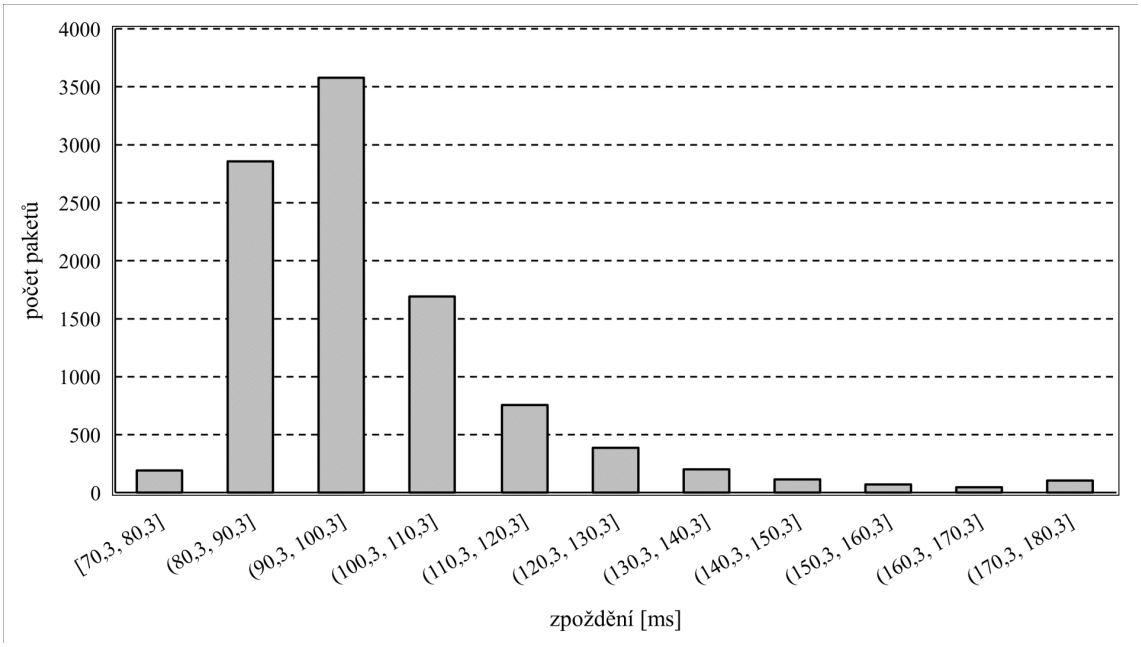
Obr. 4.10: Pareto rozložení, 100 ms – naměřené hodnoty.



Obr. 4.11: Pareto rozložení, 100 ms – histogram.



Obr. 4.12: Pareto-normální rozložení, 100 ms – naměřené hodnoty.



Obr. 4.13: Pareto-normální rozložení, 100 ms – histogram.

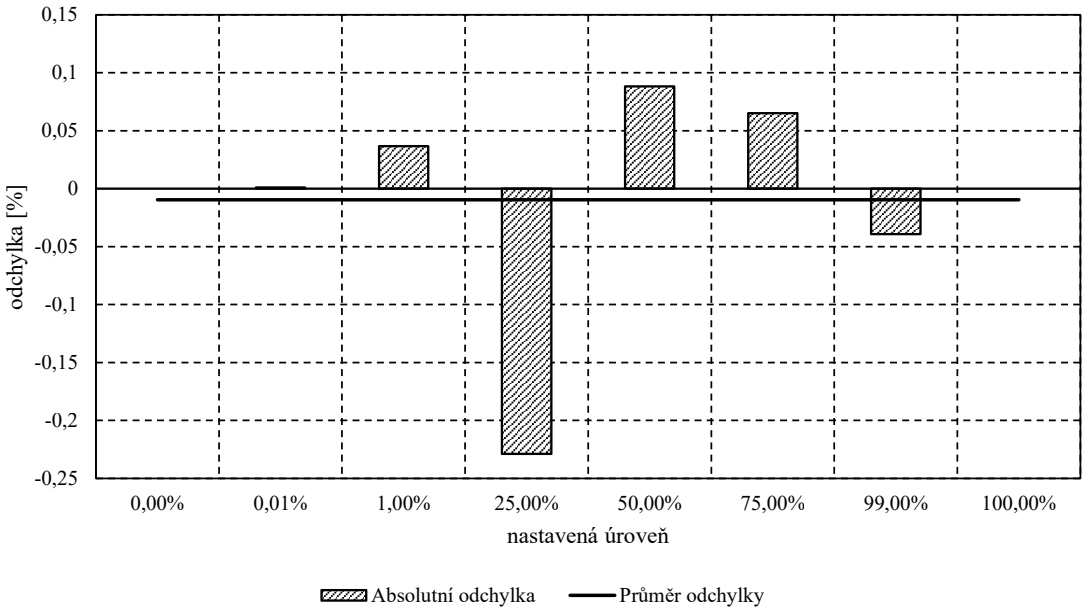
4.2.3 Změna pořadí

Měření záměny pořadí bylo provedeno pro osm úrovní záměny pořadí: 0,00 %, 0,01 %, 1,00 %, 25,00 %, 50,00 %, 75,00 %, 99,00 % a 100,00 % a pro každou úroveň byl vyslán 1 milion paketů. Procenta zaměněných paketů pro dané úrovně jsou vypsány v tabulce 4.7. V prvním sloupci tabulky jsou hodnoty požadované od emulátoru, ve druhém sloupci skutečně naměřené hodnoty a ve třetím a čtvrtém sloupci absolutní, resp. relativní, odchylka naměřených výsledků od požadovaných. Obě odchylky pro všechny hodnoty jsou vykresleny v grafech 4.14 a 4.15.

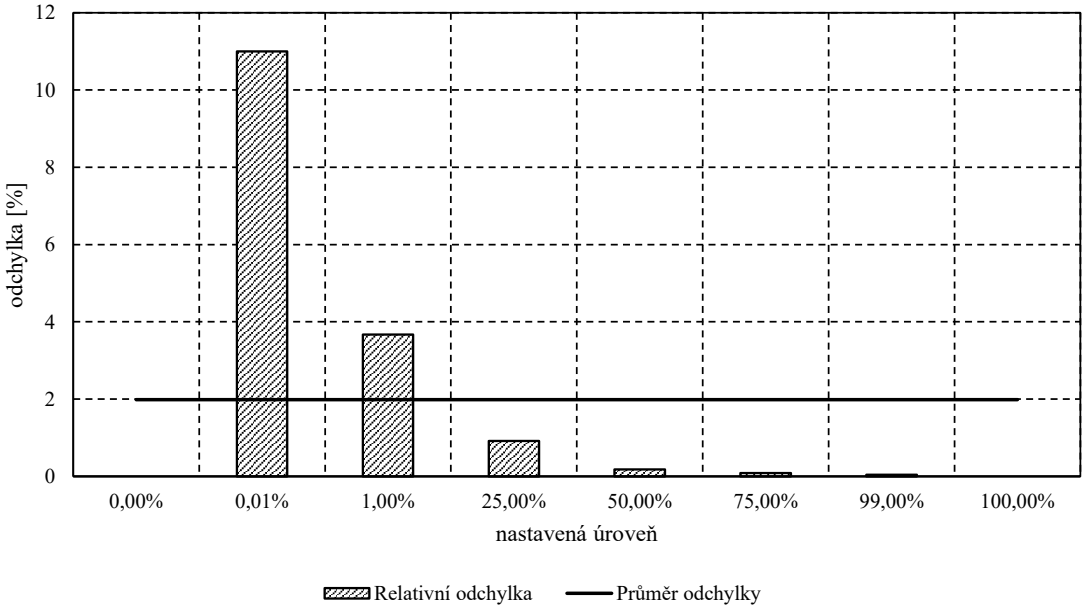
Z grafu relativní odchylky na obrázku 4.15 je patrné, že relativní odchylka dosahuje hodnoty do 1 %. Výjimkou jsou úrovně $0,01$ % a 1 %, kde relativní odchylka dosahuje 11 % a 1 %. Schopnost emulátoru emulovat záměnu pořadí je tedy u nižších hodnot záměny pořadí nižší, přesto je však výsledek emulace uspokojivý. U nejhoršího případu, úrovně $0,01$ %, došlo k změně pořadí u 111 paketů z 1 milionu, přičemž zaměněných by mělo správně být 100. Absolutní odchylka je tedy pouze 11 paketů, což je $0,0011$ %. Celková průměrná absolutní odchylka je $-0{,}0096$ % a relativní 1,7473 %.

Tab. 4.7: Výsledky měření záměny paketů.

Požadovaná úroveň [%]	Naměřená úroveň [%]	Absolutní odchylka [%]	Relativní odchylka [%]
0,00	0,00	0,00	0,00
0,01	0,01	0,00	11,00
1,00	1,04	0,04	3,67
25,00	24,77	-0,23	0,92
50,00	51,00	1,00	2,00
75,00	75,07	0,07	0,09
99,00	98,96	-0,04	0,04
100,00	100,00	0,00	0,00



Obr. 4.14: Záměna pořadí – absolutní odchylka.



Obr. 4.15: Záměna pořadí – relativní odchylka.

4.2.4 Duplikace

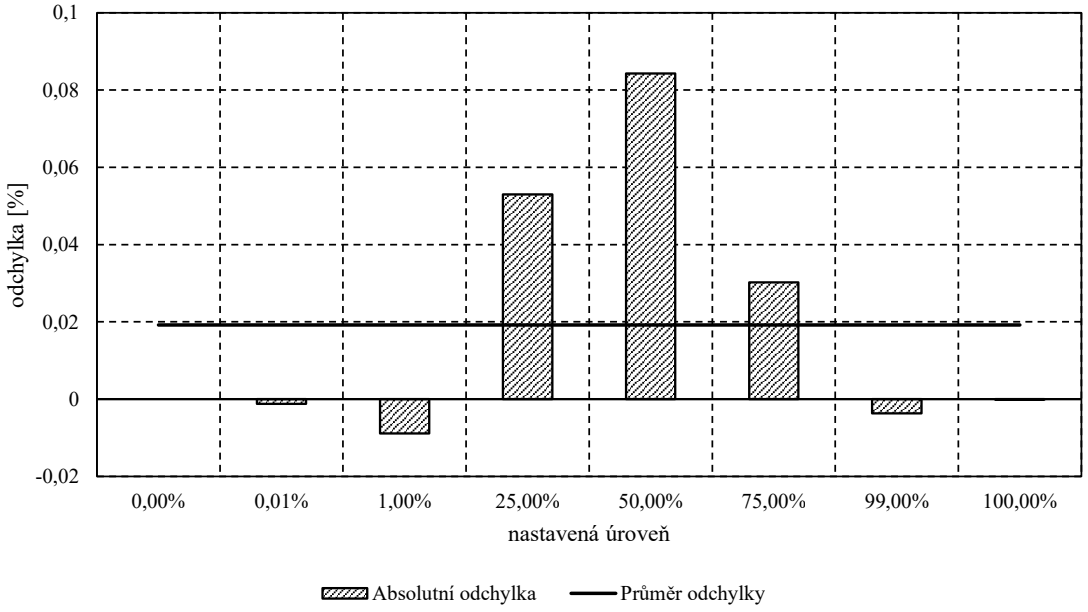
Měření schopnosti duplikovat pakety bylo provedeno stejně jako měření záměny pořadí, tj. pro každou z osmi úrovní duplikace (0,00 %, 0,01 %, 1,00 %, 25,00 %, 50,00 %,

75,00 %, 99,00 % a 100,00 %) byl vyslán 1 milion paketů. Výsledky měření jsou vypsány v tabulce 4.8. V prvním sloupci tabulky jsou, stejně jako v předcházejícím případě, hodnoty požadované od emulátoru, ve druhém sloupci skutečně naměřené hodnoty a ve třetím a čtvrtém sloupci absolutní a relativní, odchylky. Obě odchylky pro všechny hodnoty jsou vykresleny v grafech 4.16 a 4.17.

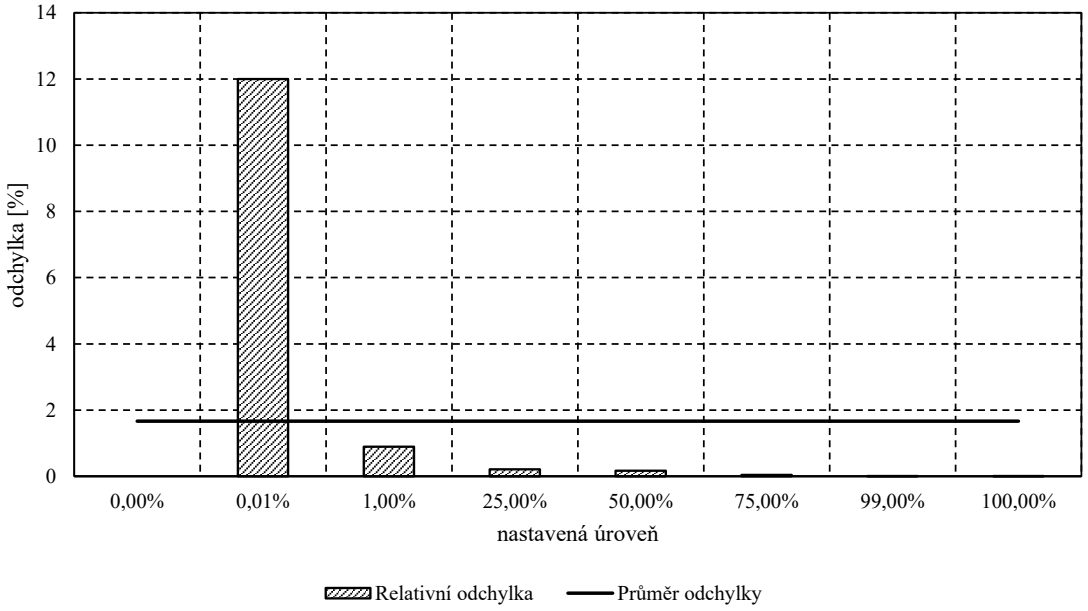
Podobně jako u záměny pořadí paketů je i u duplikace paketů nejvyšší relativní odchylka u nízkých úrovní nastavení duplikace. Původ velikosti této odchylky je stejný jako v předcházejícím případě. I malá absolutní odchylka způsobuje u malých hodnot velkou relativní odchylku. Příkladem může být opět úroveň 0,01 %, kde bylo z požadovaných 100 duplikovaných paketů skutečně duplikováno 88. Chybně tedy bylo rozhodnuto o duplikaci u 12paketů, což u celkového 1 milionu vyslaných paketů je absolutní rozdíl pouze -0,0089 %. Relativní odchylka vyšších hodnot duplikace je do 0,2120 %, celková průměrná absolutní odchylka je -0,0192 % a relativní 1,6643 %.

Tab. 4.8: Výsledky měření duplikace paketů.

Požadovaná úroveň [%]	Naměřená úroveň [%]	Absolutní odchylka [%]	Relativní odchylka [%]
0,00	0,00	0,00	0,00
0,01	0,01	0,00	12,00
1,00	0,99	-0,01	0,89
25,00	25,05	0,05	0,21
50,00	50,08	0,08	0,17
75,00	75,03	0,03	0,04
99,00	99,00	0,00	0,00
100,00	100,00	0,00	0,00



Obr. 4.16: Duplikace – absolutní odchylka.



Obr. 4.17: Duplikace – relativní odchylka.

4.2.5 Ztrátovost

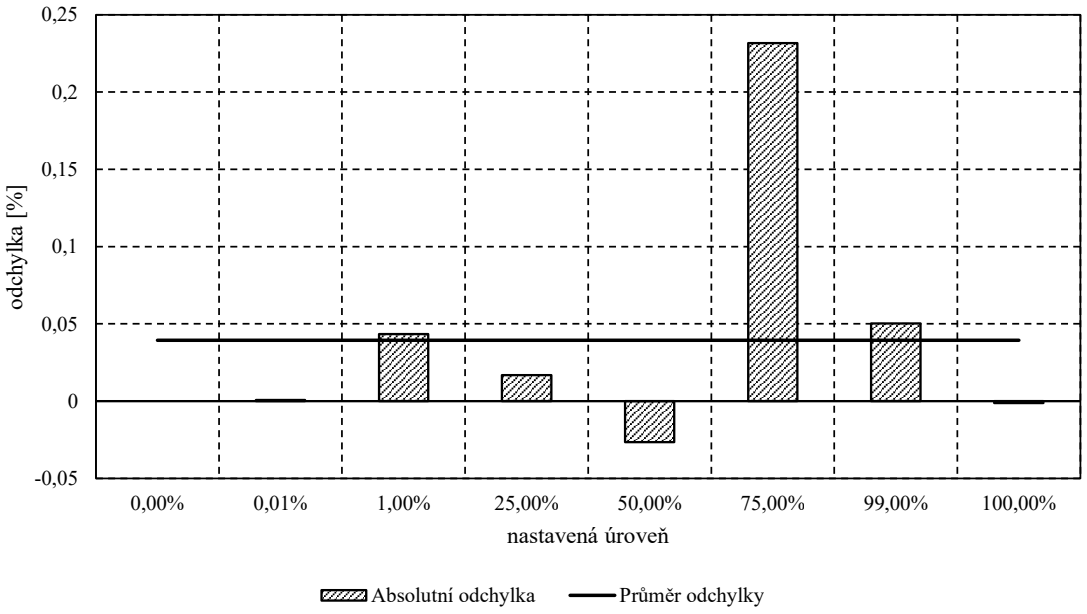
Shodně s ostatními měřeními bylo i měření ztrátovosti provedeno pro osm úrovní ztrátovosti: 0,00 %; 0,01 %; 1,00 %; 25,00 %; 50,00 %; 75,00 %; 99,00 %; 100,00 %

s 1 milionem pokusů. Výsledky měření jsou vypsány v tabulce 4.9. V prvním sloupci tabulky jsou hodnoty požadované od emulátoru, ve druhém sloupci skutečně naměřené hodnoty a ve třetím a čtvrtém sloupci absolutní, resp. relativní, odchylka naměřených výsledků od požadovaných. Obě odchylky pro všechny hodnoty jsou vykresleny v grafech 4.18 a 4.19.

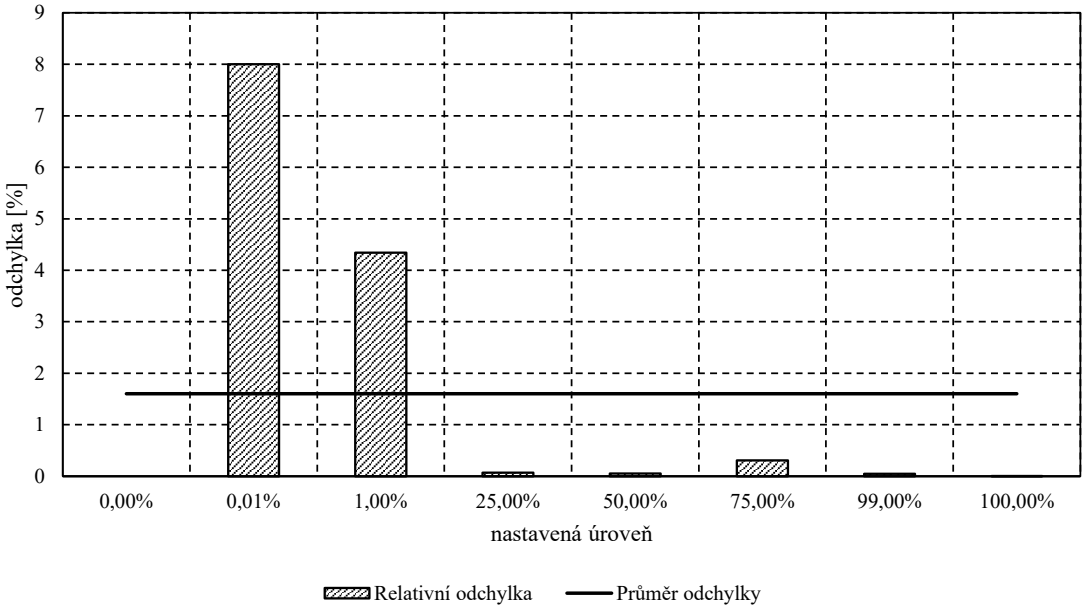
Preciznost emulace ztrátovosti je podobná jako u předcházejících parametrů. Také zde je nejnižší přesnost u nízkých nastavených hodnot ztrátovosti. U hodnot vyšších než 1 % ztrátovosti se relativní odchylka vešla pod 0,31 %. Celková průměrná absolutní odchylka je pak 0,0394 % a relativní 1,6027 %.

Tab. 4.9: Výsledky měření ztrátovosti paketů.

Požadovaná úroveň [%]	Naměřená úroveň [%]	Absolutní odchylka [%]	Relativní odchylka [%]
0,00	0,00	0,00	0,00
0,01	0,01	0,00	8,00
1,00	1,04	0,04	4,34
25,00	25,02	0,02	0,07
50,00	49,97	-0,03	0,05
75,00	75,23	0,23	0,31
99,00	98,05	0,05	0,05
100,00	100,00	0,00	0,00



Obr. 4.18: Ztrátovost – absolutní odchylka.



Obr. 4.19: Ztrátovost – relativní odchylka.

4.3 Zhodnocení naměřených výsledků

V předchozích sekcích této kapitoly byly popsány výsledky měření jednotlivých přenosových parametrů emulovaných vyvíjeným emulátorem. Jako hlavní ukazatel přes-

nosti emulace byla brána průměrná relativní odchylka každého parametru, jejichž souhrnný výpis je v tabulce 4.10.

Tab. 4.10: Souhrnné výsledky měření vyvíjeného emulátoru.

Přenosový parametr	Relativní odchylka [%]
Propustnost	8,55
Zpoždění	1,42
Proměnné zpoždění	0,15
Záměna pořadí	1,99
Duplikace	1,66
Ztrátovost	1,60

Přesnost emulovaných parametrů je závislá na výkonnosti technického vybavení. Přestože bylo měření prováděno na provizorní sestavě, byl emulátor schopen emulovat přenosové parametry s přesností do $\pm 2\,\%$. Tyto výsledky jsou srovnatelné s výsledky dosaženými v publikovaných pracích [32], [33] a [34] a lze je tedy stejně jako ve zmíněných pracích označit za uspokojivé.

Emulace propustnosti dosahovala odchylky $8,55\,\%$. Z uvedených prací se měření propustnosti věnuje pouze [33], ve které byly naměřené odchylky mnohem nižší. Přesnost emulace propustnosti je však náchylná na parametry měření více než ostatní výše zmíněné veličiny. Záleží nejen na zvoleném typu emulace a nastavení dané fronty, ale také na velikosti datových jednotek nebo na zvoleném přenosovém protokolu.

5 ZÁVĚR

Tato diplomová práce se zabývá vývojem emulátoru přenosových parametrů datových sítí. Síťové emulátory slouží k napodobování přenosových podmínek reálných prostředí v prostředích laboratorních, čehož se využívá například při testování síťových aplikací a protokolů, optimalizaci síťové infrastruktury nebo při výuce či výzkumu.

V úvodní kapitole této práce jsou kromě rozdílu mezi emulací a ostatními experimentálními technikami jmenovány jednotlivé přenosové parametry a jejich vztah k emulaci. Především u parametru propustnost panuje v jeho správném pojmenování vysoká nejednoznačnost a bylo nutné jej přesně definovat.

Před samotným vývojem zařízení byla provedena analýza současných emulátorů. Byly zkoumány jak emulátory s otevřeným kódem, tak komerční řešení. Rozbor volně dostupných emulátorů ukázal, že zkoumané, volně dostupné emulátory, jsou založeny na programech NetEm a Dummynet, jež jsou, resp. můžou být, součástí jádra operačního systému Linux. Známé emulátory (např. WANem) jsou pak jen nadstavbou nad těmito programy, kterou více či méně rozšiřují, nejčastěji o grafické uživatelské rozhraní.

Vyvinutý emulátor je součástí projektu VI20152018002 – Zátěžový tester ICT, na kterém v rámci bezpečnostního výzkumu MVČR spolupracují firma GiTy a.s. a Ústav telekomunikací FEKT VUT v Brně. Výsledný ICT tester bude kromě emulace přenosových parametrů poskytovat nástroje pro zátěžové testování a síťovou sondu a všechny tyto komponenty budou provozovány na jednom hardwarovém základu. Z tohoto důvodu je v této práci řešena pouze softwarová část emulátoru, jejíž struktura byla navržena tak, aby byla co nejvíce univerzální a bylo možné ji využít i při vývoji ostatních komponent.

Koncepce emulátoru je rozdělena na tři části, jádro emulátoru, obslužnou část a uživatelské rozhraní. Jádrem emulátoru jsou nástroje jádra operačního systému Linux Traffic Control a NetEm. Obslužná část je realizována jako API, k jejímuž vývoji byl použit programovací jazyk Java. Interakci s uživatelem zajišťuje plně klientské webové uživatelské rozhraní. Kromě deskripce softwaru emulátoru jsou ve třetí kapitole popsány také možnosti zapojení emulátoru, potažmo celého testeru, do počítačové sítě.

Přesnost emulovaných parametrů je závislá na výkonnosti technického vybavení a parametrech prováděného měření. Přestože bylo měření realizováno na provizorní sestavě, byl emulátor schopen emulovat přenosové parametry s přesností do 2 %. Výjimkou byla emulace propustnosti, která je na hardwaru a zvolených parametrech měření závislá nejvíce. Přesnost emulace propustnosti dosahovala cca $8,55\,\%$.

LITERATURA

[1] BEURAN, Razvan. *Introduction to network emulation*. Singapore: Pan Stanford Pub, 2013. ISBN 98-143-6409-6.

[2] IMRAN, Muhammad, Abas Md SAID a Halabi HASBULLAH. A survey of simulators, emulators and testbeds for wireless sensor networks. In: *2010 International Symposium on Information Technology* [online]. IEEE, 2010, 897-902 [cit. 2016-04-15]. ISBN 978-1-4244-6715-0.

[3] GURUPRASAD, Shashi; RICCI, Robert; LEPREAU, Jay. Integrated network experimentation using simulation and emulation. In: *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*. IEEE, 2005. s. 204-212.

[4] BLUM, Richard. *Network performance open source toolkit: using Netperf, tcptrace, NIST Net, and SSFNet*. Indianapolis, Ind.: Wiley Pub., 2003, xxiii, 405 s.

[5] COMER, Douglas. *Computer networks and internets*. Páté. New Jersey: Pearson Education, Inc., 2009. ISBN 0-13-606127-3.

[6] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS*. 3. aktualiz. a rozš. vyd. Praha: Computer Press, 2002, xiv, 542 s. Všechny cesty k informacím. ISBN 80-722-6675-6.

[7] PUŽMAN, Josef a J PUŽMAN. *Dálkový přenos dat*. 2. přeprac. vyd. Praha Bratislava: SNTL, ALFA, 1985, 507 s.

[8] PETERSON, Larry L a Bruce S. DAVIE. *Computer networks: a system approach*. San Francisco: Morgan Kaufmann, 1996, xxiii, 550 s. ISBN 1-55860-368-9.

[9] FEIBEL, Werner. *Encyklopedie počítačových sítí*. Praha: Computer Press, 1996, 1230 s. ISBN 80-85896-67-2

[10] PUŽMANOVÁ, Rita. *Moderní komunikační sítě*. Praha: Computer Press, 1998, 446 s. ISBN 80-7226-098-7.

[11] KUROSE, James F a Keith W. ROSS. *Computer networking: a top-down approach*. 6th ed., International. Boston ; London: Pearson, 2013, 888 s. ISBN 978-0-273-76896-8.

[12] HURA, G. S a M SINGHAL. *Data and computer communications*. New York: CRC Press, 2001, 1140 s. ISBN 0-8493-0928-X.

[13] *Cisco Networking Academy Program: CCNA 1 and 2 companion guide*. 3rd ed. Indianapolis, Ind.: Cisco, 2003, xxxiii, 1079 s. ISBN 15-871-3110-2.

[14] Přednáška: Počítačové sítě, verze 4.0, lekce č. 4, náhled slidů. *EArchiv: Archiv článků a přednášek Jiřího Peterky* [online]. [cit. 2016-04-15]. <http://www.earchiv.cz/l226/nahled.php3?l=4&me=1>.

[15] MCCABE, James D. *Practical computer network analysis and design*. San Francisco, Calif.: Morgan Kaufmann Publishers, 1998, xxii, 367 p. ISBN 15-586-0498-7.

[16] JEŘÁBEK, Jan. *Komunikační technologie*. Brno: FEKT VUT v Brně, 2013. ISBN 978-80-214-4713-4.

[17] ČÍKA, Petr. *Multimediální služby*. Brno: FEKT VUT v Brně, 2012. ISBN 978-80-214-4443-0.

[18] Terminology. *Linux Advanced Routing & Traffic Control HOWTO* [online]. [cit. 2016-04-17]. <http://lartc.org/howto/lartc.qdisc.terminology.html>.

[19] Classless Queuing Disciplines (qdiscs). *The Linux Documentation Project* [online]. [cit. 2016-04-17]. <http://www.tldp.org/HOWTO/Traffic-Control-HOWTO/classless-qdiscs.html>.

[20] SUNDARAPANDIAN, V. *Probability, statistics and queuing theory.*. Eastern economy ed. New Delhi: PHI Learning, 2009. ISBN 978-812-0338-449.

[21] KOTON, Jaroslav. *Moderní síťové technologie*. Brno: FEKT VUT v Brně, 2014. ISBN 978-80-214-5026-4.

[22] KOBLÍŽEK, Michal. *Zajištění QoS v rozsáhlých sítích.*. Brno, 2008. Dostupné také z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=6601>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústavu automatizace a informatiky. Vedoucí práce Prof. RNDr. Ing. Jiří Šťastný, CSc.

[23] NIST Net Home Page. *NIST/ITL Advanced Network Technologies Division (ANTD,892)* [online]. [cit. 2016-04-17]. <Dostupnéz:http://www-x.antd.nist.gov/nistnet/>.

[24] HEMMINGER, Stephan. In: *Network Emulation with NetEm* [online]. Canberra, Australia: Open Source Development Labs, 2005, 9 s. [cit. 2015-12-03]. Proceedings of the 6th Australia's National Linux Conference (LCA2005).

Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.1687&rep=rep1&type=pdf>.

[25] LUDOVICI, Fabio a Hagen Paul PFEIFER. NETEM(8). *Man7.org* [online]. 2011 [cit. 2015-12-03]. Dostupné z: <Dostupnéz:http://www.linuxfoundation.org/collaborate/workgroups/networking/netem>.

[26] Netem. *The Linux Foundation* [online]. [cit. 2016-04-17]. Dostupné z: <http://lartc.org/manpages/tc.txt>.

[27] HUBERT, Bert. TC(8). *Lartc.org* [online]. 2001 [cit. 2015-12-03]. Dostupné z: <http://lartc.org/manpages/tc.txt>.

[28] TATA CONSULTANCY SERVICES (TCS). *WANem 2.0: Wide Area Network Emulator* [online]. 2008, 34 s. [cit. 2015-12-03]. Dostupné z: <http://sourceforge.net/projects/wanem/files/Documents/wanemulator_all_about_v2.0.pdf/download?use_mirror=skylink>.

[29] RIZZO, Luigi. Dummynet. *ACM SIGCOMM Computer Communication Review*. 1997, 27(1): 31-41. DOI: 10.1145/251007.251012. ISSN 01464833. Dostupné také z: <http://portal.acm.org/citation.cfm?doid=251007.251012>.

[30] CARBONE, Marta a Luigi RIZZO. *Dummynet Revisited* [online]. Universita‘ di Pisa, Dipartimento di Ingegneria dell'Informazione, 2009, 8 s. [cit. 2005-12-03]. Dostupné z: <http://info.iet.unipi.it/~luigi/papers/20091201-dummynet.pdf>.

[31] IPFW(8). THE FREEBSD PROJECT. *Freebsd.org* [online]. 2012 [cit. 2015-12-03]. Dostupné z: <https://www.freebsd.org/cgi/man.cgi?query=ipfw&manpath=FreeBSD+9-current&format=html>.

[32] JURGELIONIS, Audrius, Jukka-Pekka LAULAJAINEN, Matti HIRVONEN a Alf Inge WANG. An Empirical Study of NetEm Network Emulation Functionalities. *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)* [online]. IEEE, 2011, : 1-6 [cit. 2015-12-10]. DOI: 10.1109/ICCCN.2011.6005933. ISBN 978-1-4577-0637-0. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6005933>.

[33] Nussbaum, Lukas a Oliver RICHARD. A Comparative Study of Network Link Emulators. *Proceedings of the 2009 Spring Simulation Multiconference* [online].

SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009. Dostupné z: <http://portal.acm.org/citation.cfm?id=1639809.1639898>.

[34] LUBKE, Robert, Peter BUSCHEL, Daniel SCHUSTER a Alexander SCHILL. Measuring accuracy and performance of network emulators. *2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)* [online]. IEEE, 2014, : 63-65 [cit. 2015-12-10]. DOI: 10.1109/BlackSeaCom.2014.6849005. ISBN 978-1-4799-4067-7. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6849005>.

[35] NANNI, Dan. How to install dummynet on CentOS. *Xmodulo.com* [online]. 2013 [cit. 2015-12-10]. Dostupné z: <http://xmodulo.com/how-to-install-dummynet-on-centos.html>.

[36] JAR TECHNOLOGIES LTD. *JAR Technologies Leading Enterprise Software Testing Tools: Overview - JAR Technologies* [online]. 2015 [cit. 2015-12-10]. Dostupné z: <http://jartechnologies.com/networkemulators-overview>.

[37] APPOSITE TECHNOLOGIES, INC. *Apposite Technologies :: The Leader In WAN Emulation: Apposite Technologies :: Netropy 40G WAN Emulator* [online]. 2015 [cit. 2015-12-10]. Dostupné z: <http://www.apposite-tech.com/products/netropy-40G.html>.

[38] ITRINEGY LTD. *Network Emulation | Application Performance Monitoring | APM: 50Mbps, 100Mbps, 200Mbps and 1Gbps Network Emulators from iTrinegy* [online]. 2015 [cit. 2015-12-10]. Dostupné z: <http://www.itrinegy.com/index.php/products/network-emulators/ne-one>.

[39] ZTI COMMUNICATIONS. *Xmodulo.com* [online]. 2013 [cit. 2015-12-10]. Dostupné z: <http://www.zti-communications.com/netdisturb>.

[40] SOFTPERFECT PTY LTD. *OftPerfect : software for networks, enterprises and developers: SoftPerfect WAN Connection Emulator for Windows* [online]. 2013 [cit. 2015-12-10]. Dostupné z: <https://www.softperfect.com>.

[41] The DCI Architecture: A New Vision of Object-Oriented Programming. *The Artima Developer Community* [online]. 2013 [cit. 2016-04-18]. Dostupné z: <http://www.artima.com/articles/dci_vision.html>.

[42] NANNI, Dan. How to install dummynet on CentOS. *Xmodulo.com* [online]. 2013 [cit. 2015-12-10]. Dostupné z: <http://xmodulo.com/how-to-install-dummynet-on-centos.html>.

[43] *Apache Tomcat: Welcome!* [online]. 2013 [cit. 2015-4-23]. Dostupné z: <https://tomcat.apache.org>.

[44] *Maven: Welcome to Apache Maven* [online]. 2013 [cit. 2015-4-23]. Dostupné z: <https://maven.apache.org>.

[45] Spring Framework. *Spring* [online]. 2013 [cit. 2015-4-23]. Dostupné z: <https://projects.spring.io/spring-framework>.

[46] Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedry fyziky. *Chuby měření* [online]. [cit. 2016-04-24]. Dostupné z: <www.kfy.zcu.cz/dokumenty/FP1/chyby_mereni.pdf>.

[47] Určování chyb měřicích přístrojů. *Úvodník k počítači Amper* [online]. [cit. 2016-04-24]. Dostupné z: <http://amper.ped.muni.cz/jenik/nejistoty/html_tree/node14.htm>.

[48] Přesnost a chyby měření. *Fyzikální praktikum – elektronická podpora výuky* [online]. [cit. 2016-04-24]. Dostupné z: <http://home.pf.jcu.cz/~kriz/index.php?option=com_content&view=article&id=226:chybymer&catid=55:fpr1&Itemid=27>.

[49] Iperf3: iperf3 3.1.2 documentation. *ESnet Software: ESnet Software 1.0 documentation* [online]. [cit. 2016-04-24]. Dostupné z: <http://software.es.net/iperf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Aplikační rozhraní – Application Programming Interface
ADSL	Asymmetric Digital Subscriber Line
ARP	Address Resolution Protocol
BER	bitová chybovost – Bit Error Rate
BSD	Berkeley Software Distribution
CBQ	Class Based Queueing
CSS	kaskádové styly – Cascading Style Sheets
DHCP	protokol pro automatickou konfiguraci uzlů v síti – Dynamic Host Configuration Protocol
FIFO	fronta se systémem: první dovnitř, první ven – First In First Out
GNU GPL	všeobecná veřejná licence GNU – GNU General Public License
GUI	grafické uživatelské rozhraní – Graphical User Interface
HTB	Hierarchical Token Bucket
HTML	značkovací jazyk pro tvorbu webových stránek – HyperText Markup Language
HTTP	protokol pro výměnu hypertextových dokumentů – HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IP	protokol síťové vrstvy – Internet Protocol
IPC	meziprocesová komunikace – Inter-Process Communication
IPTV	protokol pro přenos televizního vysílání přes Internet – Internet Protocol Television
ISO/OSI	síťový referenční model ISO/OSI – International Organization for Standardization/Open Systems Interconnection model
ISP	poskytovatel internetového připojení – Internet Service Provider
LAN	lokální sítě – Local Area Network

MAN	metropolitní sítě – Metropolitan Area Network
MVC	návrhový vzor MVC – Model-View-Controller
NAT	překlad síťových adres – Network Address Translation
OS	operační systém
PAN	personální sítě – Personal Area Network
PFIFO	Packet limited First In, First Out queue – Packet limited First In, First Out fronta
PHP	hypertextový preprocesor
PID	identifikátor procesu v operačním systému – Process Identifier
RTT	obousměrné zpoždení – Round-Trip-Time
TCP	protokol transportní vrstvy – Transmission Control Protocol
TCP/IP	referenční model TCP/IP – Transmission Control Protocol/Internet Protocol
UDP	protokol transportní vrstvy – User Datagram Protocol
UI	uživatelské rozhraní – User Interface
UTP	nestíněná kroucená dvoulinka – Unshielded Twisted Pair
VoD	video na vyžádání – Video on Demand
VoIP	hlas přes Internet – Voice over Internet Protocol
VVoIP	hlas i video přes Internet – Voice and Video over Internet Protocol
VPN	virtuální privátní síť – Virtual Private Network
WAN	globální sítě – Wide Area Network
WUI	webové uživatelské rozhraní – Web User Interface

SEZNAM PŘÍLOH

A KONFIGURACE EMULÁTORU PRO JEDNOTLIVÉ SCÉNÁŘE IMPLEMENTACE

A.1 Konfigurace testovacího stroje a globální nastavení

Konfigurace umožňující aplikaci jednotlivých scénářů byla prováděna na stroji s dvěma síťovými rozhraními, viz tabulka A.1 a operačním systémem CentOS 7.2-1511, s jádrem Linux 3.10.0-327.

Tab. A.1: Síťová rozhraní testovacího stroje.

Název rozhraní	Popis rozhraní
enp1s0	rozhraní samostatné síťové karty
enp4s0	rozhraní integrované síťové karty

Pro získání aktuální verze operačního systému lze využít příkaz `yum update` A.1.

Výpis A.1: Příkaz pro získání aktuální verze.

```
1 # yum update
```

Pro aplikaci změn provedených v konfiguračních souborech síťových rozhraní je nutné jejich restartování pomocí příkazů `ifdown` a `ifup` (ukázka pro rozhraní enp1s0).

Výpis A.2: Příkaz pro aplikaci změn síťových rozhraní.

```
1 # ifdown enp1s0
2 # ifup enp1s0
```

A.1.1 Instalace a spuštění DHCP služby

Výpis A.3: Instalace a spuštění DHCP služby.

```
# yum install dhcp
# systemctl enable dhcpd
# systemctl start dhcpd
(# systemctl restart dhcpd)
```

A.1.2 Vypnutí služby NetworkManager

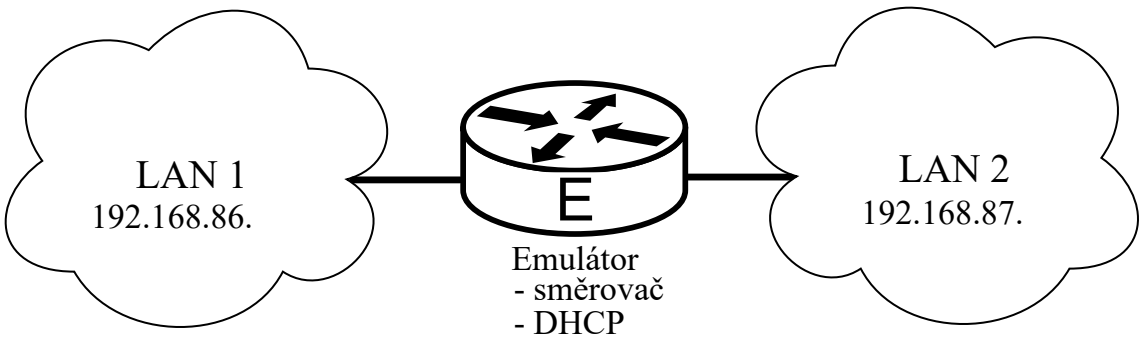
Výpis A.4: Vypnutí služby NetworkManager.

```
# systemctl stop NetworkManager
# systemctl disable NetworkManager
```

A.2 Konfigurace testovacího stroje pro navržené scénáře zapojení emulátoru

A.2.1 Plně autonomní prostředí - směrovač

Schéma topologie:



Výpis A.5: Plně autonomní prostředí S – /etc/sysconfig/network-scripts/enp1s0.

```
# ifcfg-enp1s0

TYPE=Ethernet
BOOTPROTO=static
NM_CONTROLLED=no
NAME=enp1s0
UUID=8310bdba-207d-4165-bed4-74ea643eb95b
DEVICE=enp1s0
ONBOOT=yes
IPADDR=192.168.86.1
NETMASK=255.255.255.0
```

Výpis A.6: Plně autonomní prostředí S – /etc/sysconfig/network-scripts/enp4s0.

```
# ifcfg-enp4s0

TYPE=Ethernet
BOOTPROTO=static
NM_CONTROLLED=no
NAME=enp4s0
UUID=90d756c7-0f3c-4443-8d82-b8eb4c850ebc
DEVICE=enp4s0
ONBOOT=yes
IPADDR=192.168.87.1
NETMASK=255.255.255.0
```

Výpis A.7: Plně autonomní prostředí S – /etc/dhcp/dhcpd.conf.

```
# dhcpd.conf
# emulator

option domain-name "emulator.test";
option domain-name-servers 8.8.8.8;

default-lease-time 600;
max-lease-time 7200;

ddns-update-style interim;

authoritative;

ignore client-updates;

set vendorclass = option vendor-class-identifier;

# Use this to send dhcp log messages to a different log file (you
    also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

subnet 192.168.86.0 netmask 255.255.255.0 {
  option domain-name    "enp1s0.emulator.test";
  interface       enp1s0;
  range       192.168.86.101 192.168.86.110;
  option routers      192.168.86.1;
}

subnet 192.168.87.0 netmask 255.255.255.0 {
  option domain-name    "enp4s0.emulator.test";
  #interface      enp4s0;
  range       192.168.87.101 192.168.87.110;
  option routers      192.168.87.1;
}
```

Výpis A.8: Plně autonomní prostředí S – /etc/sysctl.d/99-sysctl.conf.

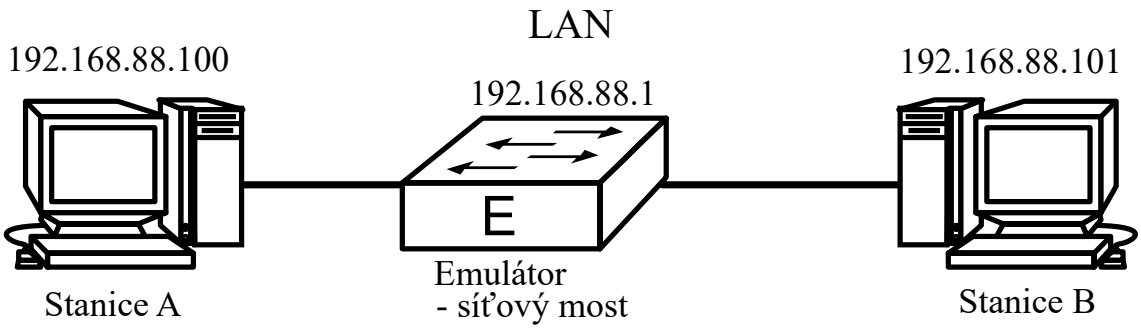
```
net.ipv4.ip_forward=1
```

Zdroje:

Linux Router and Firewall. *Bromosapien.net* [online]. 20144 [cit. 2016-02-03]. Dostupné z: <`https://www.bromosapien.net/media/index.php/Linux_Router_and_Firewall`>.

A.2.2 Plně autonomní prostředí - síťový most

Schéma topologie:



Výpis A.9: Příkaz pro naistalaci balíčku bridge-utils.

```
# yum install bridge-utils
```

Výpis A.10: Plně autonomní prostředí B – /etc/sysconfig/network-scripts/enp1s0.

```
# ifcfg-enp1s0

TYPE=Ethernet
BOOTPROTO=none
NM_CONTROLLED=no
NAME=enp1s0
UUID=8310bdba-207d-4165-bed4-74ea643eb95b
DEVICE=enp1s0
ONBOOT=yes
BRIDGE=br0
```

Výpis A.11: Plně autonomní prostředí B – /etc/sysconfig/network-scripts/enp4s0.

```
# ifcfg-enp4s0

TYPE=Ethernet
BOOTPROTO=none
NM_CONTROLLED=no
NAME=enp4s0
UUID=90d756c7-0f3c-4443-8d82-b8eb4c850ebc
DEVICE=enp4s0
ONBOOT=yes
BRIDGE=br0
```

Výpis A.12: Plně autonomní prostředí B – /etc/sysconfig/network-scripts/br0.

```
# ifcfg-br0

TYPE=Bridge
BOOTPROTO=static
NM_CONTROLLED=no
NAME=br0
DEVICE=br0
ONBOOT=yes
IPADDR=192.168.88.1
NETMASK=255.255.255.0
STP=on
```

Výpis A.13: Plně autonomní prostředí B – /etc/dhcp/dhcpd.conf.

```
# dhcpd.conf
# emulator

option domain-name "emulator.test";
option domain-name-servers 8.8.8.8;

default-lease-time 600;
max-lease-time 7200;

ddns-update-style interim;

authoritative;

ignore client-updates;

set vendorclass = option vendor-class-identifier;

# Use this to send dhcp log messages to a different log file (you
    also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

subnet 192.168.88.0 netmask 255.255.255.0 {
  option domain-name    "br0.emulator.test";
  #interface      br0;
  range       192.168.88.101 192.168.88.110;
  option routers      192.168.88.1;
}
```

Výpis A.14: Vytvoření síťového mostu.

```
# brctl addbr br0
# brctl.addif br0 enp1s0
# brctl addif br0 enp4s0
(# ifdown br0)
# ifup br0
```

Výpis A.15: Další pomocné příkazy balíčku bridge-utils.

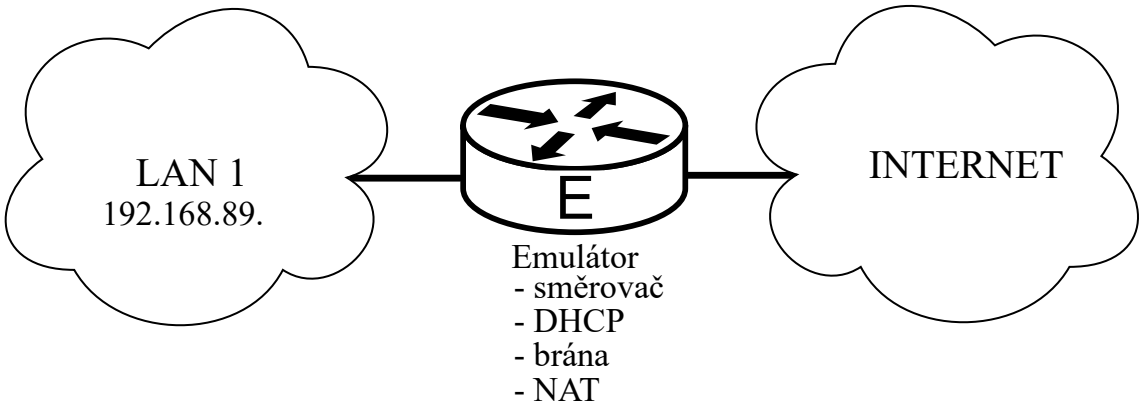
```
# brctl addbr br0
# brctl.addif br0 enp1s0
# brctl addif br0 enp4s0
(# ifdown br0)
# ifup br0
```

Zdroje:

Tom. *Tdistler.com* [online]. 2011 [cit. 2016-02-03]. Dostupné z: <`http://tdistler.com/2011/06/10/netem-wan-emulation-how-to-setup-a-netem-box`>.

A.2.3 Autonomní prostředí s přístupem do jiné sítě

Schéma topologie:



Výpis A.16: Autonomní prostředí s přístupem do jiné sítě – /etc/sysconfig/network-scripts/enp1s0.

```
# ifcfg-enp1s0

TYPE=Ethernet
BOOTPROTO=static
NM_CONTROLLED=no
NAME=enp1s0
UUID=8310bdba-207d-4165-bed4-74ea643eb95b
DEVICE=enp1s0
ONBOOT=yes
IPADDR=192.168.89.1
NETMASK=255.255.255.0
```

Výpis A.17: Autonomní prostředí s přístupem do jiné sítě – /etc/sysconfig/network-scripts/enp4s0.

```
# ifcfg-enp4s0

TYPE=Ethernet
BOOTPROTO=dhcp
NM_CONTROLLED=no
NAME=enp4s0
UUID=90d756c7-0f3c-4443-8d82-b8eb4c850ebc
```

Výpis A.18: Autonomní prostředí s přístupem do jiné sítě – /etc/dhcp/dhcpd.conf.

```
# dhcpd.conf
# emulator

option domain-name "emulator.test";
option domain-name-servers 8.8.8.8;

default-lease-time 600;
max-lease-time 7200;

ddns-update-style interim;

authoritative;

ignore client-updates;

set vendorclass = option vendor-class-identifier;

# Use this to send dhcp log messages to a different log file (you
    also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

subnet 192.168.89.0 netmask 255.255.255.0 {
  option domain-name    "gw.emulator.test";
  range       192.168.89.101 192.168.89.110;
  option routers      192.168.89.1;
}
```

Výpis A.19: Autonomní prostředí s přístupem do jiné sítě – /etc/sysctl.d/99-sysctl.conf.

```
net.ipv4.ip_forward=1
```

Výpis A.20: Instalace služby iptables.

```
# yum install iptables-services iptables-utils
```

Výpis A.21: Služba firewall.

```
# systemctl stop firewalld
# systemctl disable firewalld

# systemctl enable iptables
# systemctl start iptables
```

Výpis A.22: Autonomní prostředí s přístupem do jiné sítě – /etc/sysconfig/iptables.

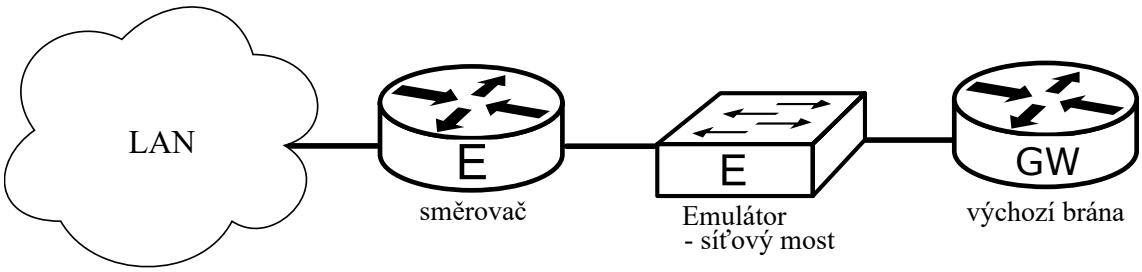
```
# iptables

*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o enp4s0 -j MASQUERADE
COMMIT
}
```

Zdroje:

Linux Router and Firewall. *Bromosapien.net* [online]. 20144 [cit. 2016-02-03]. Dostupné z: <`https://www.bromosapien.net/media/index.php/Linux_Router_and_Firewall`>.

A.2.4 Připojení mezi dva uzly sítě

Schéma topologie:



Výpis A.23: Příkaz pro instalaci balíčku bridge-utils.

```
# yum install bridge-utils
```

Výpis A.24: Připojení mezi dva uzly sítě – /etc/sysconfig/network-scripts/enp1s0.

```
# ifcfg-enp1s0

TYPE=Ethernet
BOOTPROTO=none
NM_CONTROLLED=no
NAME=enp1s0
UUID=8310bdba-207d-4165-bed4-74ea643eb95b
DEVICE=enp1s0
ONBOOT=yes
BRIDGE=br0
```

Výpis A.25: Připojení mezi dva uzly sítě – /etc/sysconfig/network-scripts/enp4s0.

```
# ifcfg-enp4s0

TYPE=Ethernet
BOOTPROTO=none
NM_CONTROLLED=no
NAME=enp4s0
UUID=90d756c7-0f3c-4443-8d82-b8eb4c850ebc
DEVICE=enp4s0
ONBOOT=yes
BRIDGE=br0
```

Výpis A.26: Připojení mezi dva uzly sítě – /etc/sysconfig/network-scripts/br0.

```
# ifcfg-br0

TYPE=Bridge
BOOTPROTO=none
NM_CONTROLLED=no
DEVICE=br0
ONBOOT=yes
```

Zdroje:

Tom. *Tdistler.com* [online]. 2011 [cit. 2016-02-03]. Dostupné z: <http://tdistler.com/2011/06/10/netem-wan-emulation-how-to-setup-a-netem-box>.

B TESTOVACÍ PROGRAM PRO MĚŘENÍ ZÁMĚNY POŘADÍ

B.1 Klientská část

Skript generující pakety:

Výpis B.1: Skript pro testování záměny pořadí - klient.

```
#!/bin/bash
# loop for i, where i is from 1 to 1000000 with step 1
for i in $(seq 1 1 1000000)
do
# creates udp packet with data = i and sent it via netcat program
   to server
echo -n $i | netcat -u -w0 192.168.88.104 888
done
```

B.2 Serverová část

Příkaz pro extrakci dat ze zachyceného paketu v souboru s příponou pcap:

Výpis B.2: Skript pro testování záměny pořadí - server.

```
# tshark -r input.pcap -T fields -e data.data > output.txt
```

C OBSAH PŘILOŽENÉHO CD

Na přiloženém médiu se nachází

- kompletní elektronická verze práce ve formátu PDF,
- vyvinutý software emulátoru (detailněji níže C.1),
- dokumentace softwaru emulátoru (detailněji níže C.2),
- testovací program pro měření záměny pořadí,
- videoukázka emulátoru ve formátu AVI (detailněji níže C.3).

Výpis adresářové struktury přiloženého média.

- `/` ..... kořenový adresář přiloženého CD
  - `dokumentace` ..... dokumentace softwaru emulátoru
    - `doxygen`
  - `emulator` ..... vyvinutý software emulátoru
    - `program`
      - `webapi`
      - `wui`
  - `test_zameny`
    - `klient.sh`
    - `server.sh`
  - `Emulator_prenosovych_parametru_datovych_siti.pdf` ..... kompletní elektronická verze práce ve formátu PDF
  - `Emulator.avi` ..... videoukázka emulátoru ve formátu AVI

C.1 Software emulátoru

V příloze ve složce `emulator` jsou zdrojové kódy programu. Pro úspěšnou kompilaci a provoz vytvořeného softwaru jsou zapotřebí tyto prerekvizity:

- počítač se dvěma sítovými rozhraními,
- operační systém Linux CentOS 7,
- technologie Java verze 8,
- webový server Apache Tomcat verze 9,
- nástroj pro správu sestavení Maven.

Webové uživatelské rozhraní lze spustit v běžném internetovém prohlížeči, ovšem jeho funkčnost je bez fungující API značně omezena.

C.2 Dokumentace softwaru emulátoru

Dokumentace k programu emulátoru byla provedena nástrojem Doxygen. HTML výstup z tohoto nástroje je obsahem této přílohy. Pro prohlížení dokumentace stačí

otevřít soubor index.html. Je však nutné mít na počítači nainstalovaný prohlížeč souborů HTML, například Internet Explorer.

C.3 Videoukázka emulátoru

Jelikož je instalace softwaru emulátoru relativně složitá a vyžaduje mnoho prerekvizit, je na přiloženém CD nosiči také videoukázka funkčnosti emulátoru ve formátu AVI. Pro omezenou velikost příloh v IS VUT v Brně nemohl být výše zmíněný soubor do systému IS nahrán, a nachází se tak pouze na přiloženém médiu. Pro přehrání videa by měl být postačující běžný přehrávač videa.